8-8-1990

Transputer-based robot controller

Wei-Chieh Chang

Rochester Institute of Technology
College of Engineering
Department of Computer Engineering

Transputer-Based Robot Controller

by

Wei-Chieh Chang

A thesis, submitted to the Faculty of the Department of Computer Engineering, in partial fulfillment of the requirements for the degree of Master of Science in Computer Engineering.
Approved by:

8/8/'90

Dr. Tony H. Chang, date
Thesis Advisor

8/8/90

Dr. Roy S. Czernikowski, Professor, date
Department Head

8/8/90

Prof. George A. Brown date

Title of thesis " Transputer-Based Robot Controller",

I

hereby grant permission to the Wallaee Memorial Library of RIT to reproduce my thesis i

part or in whole. Any reproduction will not be for commercial use or profit.

Date Aug. 8, 1990

TABLE OF CONTENTS

ABSTRACT

A cost-effective architecture for the control of robot manipulators based on functional decomposition of the equations of motion is described. The Lagrange-Euler(LE) and the Newton-Euler(NE) formulations are used for decomposition. According to real-time control criterion, the LE equations are not suitable for implementation using currently available hardware because the required number of computations is too high, even after taking the inherent parallelism into account. However, the recursive nature of the Newton-Euler equations of motion lend themselves to being decomposed to terms used to generate the recursive forward and backward formulations.

A special architecture implemented on a network of transputers is proposed which takes advantage of both the parallelism and seriallism of the NE equations and the ease of building communication channel provided by the transputers and Occam language. This proposed controller model can be best defined as a macro level pipeline. Based on this model, both floating point computation and fixed point computation results are presented for performance comparison .

I. Introduction

As more advanced VLSI technologies come to the world, the fast decrease of computational costs and increase in computation power pave a smooth way to utilizing multiple microprocessors for real-time applications, such as controlling robot manipulators. Although current industrial manipulators are multiple microprocessor-based systems, they do not utilize the complete dynamic model of the manipulator in designing the control system. Independent joint models are used to model the links of the manipulator, and the controller design is based on these joint equations. This, in effect, assumes that the joints of the manipulators are decoupled, and the coupling terms, such as coriolis and centrifugal forces/torques, are negligible. As a result, they move at high speed with "noticeable" vibrations.

The main problem in using microprocessors for the real-time control of manipulators is the complexity of computations required to compute a control law based on the dynamic model. Luh and Lin[1] proposed an architecture for executing the inverse plant dynamic equations based on the Newton-Euler(NE) equations, in which they assigned a processor to calculate the joint torque for each joint using a minimum scheduling algorithm, based on branch-and-bound method. However, they did not take into account the recursive structure of the NE formulation; especially the sequential dependencies of the algorithm which are conductive to pipeling.

Orin[2], also recognizing the recursive nature of NE formulation; proposed a

pipelined architecture for computing the inverse plant and indicated that by using three CPUs arranged in pipelined fashion, a gain of almost three times in the servo sampling time could be achieved.

The objective of this thesis is to extend these ideas and to find a cost-effective architecture using transputers to perform the real-time control of a manipulator. This controller functions as a peripheral or an attached processor controller(APC) to some host computer(IBM PC-AT) which provides both software and hardware environments.

The real-time robot-arm-control problem was tackled in two separate but coherent phases of design. First, the dynamic model of a manipulator was functionally decomposed into several " computational modules." Second, based on these models, an appropriate computational structure using present day microprocessors was designed to reduce the computational time as much as possible. So the NE equations of motion is decomposed into the terms used to generate the recursive forward and backward equations. And then an architecture that meets our design criterion is proposed, which is based on the NE equations, because it is less computationally intensive than the LE formulation. A simple robot (see Fig. 2) is used as an example, though the architecture presented here is applicable to other manipulators with rotary joints.

II. Manipulator Dynamics and Control

A. General Description of NE equations and LE equations

Robot arm dynamics deals with the mathematical formulations of the equations of robot arm motion. The dynamic equations of motion of a manipulator are a set of mathematical equations describing the dynamic behavior of the manipulator. Such mathematical equations of motion are useful for computer simulation of the robot arm motion, the design of suitable control equations for a robot arm, and the evaluation of the kinematic design and structure of a robot arm. The purpose of manipulator control is to maintain the dynamic response of a computer-based manipulator in accordance with some pre-specified system performance and desired goals. The actual dynamic model of a robot arm can be obtained from known physical laws such as the laws of Newtonian mechanics and Lagrangian mechanics. This leads to the development of the dynamic equations of motion for the various articulated joints of the manipulator in terms of specified geometric and inertial parameters of the links. Conventional approaches like the LE [3],[4],[5]and NE [4],[6] formulations could then be applied systematically to develop the actual robot arm motion equations. Various forms of robot arm motion equations describing the rigid-body robot arm dynamics are obtained from these two formulations, such as Uicker's LE equations[7], [3], Hollerbach's Recursive-Lagrange (R-L) equations [8], Luh's NE equations[6], and Lee's generalized d'Alembert (G-D) equations [9]. These motion equations are "equivalent" to each other in the sense that they describe the dynamic

behavior of the same physical robot manipulator. However, the structures of these equations may differ as they are obtained for various reasons and purposes. For example, the derivation of the dynamic model of a manipulator based on the LE formulation is simple and systematic, this formulation provides explicit state equations for robot dynamic and can be utilized to analyze and design advanced joint-variable space control strategies; while some are obtained to achieve fast computation time in evaluating the nominal joint torques in servoing a manipulator. All these equations can be used on both solving forward dynamics problem, that is , given the desired torques/forces, the dynamic equations are used to solve the joint accelerations which are then integrated to solve for the generalized coordinates and their velocities; while the inverse dynamics problem is to compute the generalized forces and torques from the desired generalized coordinated and their first two time derivatives. Comparing with their computation complexities, only NE equations are suitable for real-time control implementation of inverse dynamics[10],[11],[6] to meet real-time cotrol requirements.. This set of equations contains both forward recursion and backward recursion that propagates the forces and moments exerted on each link from the end-effector of the manipulator to the base reference frame - backward recursion, and propagates linear velocities, angular velocities, angular acceleration, and linear accelerations at the center of mass of each link from the inertial coordinate frame to the hand coordinate frame - forward recursion. Before the work of Luh et al[6], all the inertial matrices Ii and the physical geometric parameters (ri, si, pi, p*i) are referenced to the base coordinate system. As a result, they change as the robot arm is moving. Luh et al improved the previous set of NE

equations of motion by referencing all velocities, accelerations, inertial matrices, location of center of mass of each link, and forces/moments to their own link coordinate systems and made the computation complexity of this equation linearly proportional to the number of degree of freedom of a manipulator. Based on this model, it would be very easy to build a real-time controller of a robot arm in the joint-variable space. The NE equations are listed in Appendix A.

B. D'Alembert Representation

In order to describe the translational and rotational relationships between adjacent links, Denavit and Hartenberg[12] proposed a matrix method of systematically establishing a coordinate system (body-attached frame) to each link of an articulated chain. The Denavit-Hartenberg(D-H) representation develops a 4 x 4 homogeneous transformation matrix representing each link's coordinate system at the joint with respect to the previous link's coordinate system. Applying this method, the end-effector of a robot expressed in the "hand coordinates" can be translated to and expressed in the "base coordinates" which makes up the inertial frame of this dynamic system. Every frame is determined and established on the basis of the following three rules:

1. The Z_{i-1} axis lies along the axis of motion of the i th joint.
2. The X_i axis is normal to the Z_{i-1} axis, and pointing away from it.
3. The Y_i axis completes the right-handed coordinate system as required.

The D-H representation of a rigid link depends on four geometric parameters associated with each link. These four parameters completely describe any revolve or translational joint. Referring to Fig. 3 these four parameters are defined as follows:

θ_i is the joint angle from the X_{i-1} axis to the X_i axis about the Z_{i-1} axis (using the right-hand rule).

d_i is the distance from the origin of the (i-1)th coordinate frame to the intersection of the Z_{i-1} axis with the X_i axis along the Z_{i-1} axis.

a_i is the offset distance from the intersection of the Z_{i-1} axis with the X_i axis to the origin of the i th frame along the X_i axis(or the shortest distance between the Z_{i-1} and Z_i axes).

α_i is the offset angle from the Z_{i-1} axis to the Z_i axis about the X_i axis completing the right hand rule).

The values of these parameters of two types of robot are shown in Fig. 1 and Fig. 2. After these parameters are determined, the transformation matrix of two adjacent coordinate frames can be derived as follows:

$$A_i^{i-1} = T_{z,d} T_{z,\theta} T_{x,a} T_{x,\alpha}$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c\theta_i & -s\theta_i & 0 & 0 \\ s\theta_i & c\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_i \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_i & -s\alpha_i & 0 \\ 0 & s\alpha_i & c\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using the $^{i-1}A_i$, a point P_i at rest in link i and expressed in coordinate frame with respect to coordinate system i, can be related to the coordinate

system i-1 built on link i-1 by

$$P_{i-1} = {}^{i-1}A_i P_i$$

For a rotary joint, d_i, a_i and $arfa_i$ are the joint parameters and remain constant for a robot, while theta is the joint variable that change when link i moves or revolves with respect to link i-1.

C. Computation Module of NE Equations of Motion

The recursive NE equations were computed serially in the past(Fig. 3); first all the angular velocities wi were computed, then the angular accelerations w.i, etc. In the parallel architecture(Fig. 4) proposed by Lathrop, it is obvious that some quantities can be computed in parallel(see Fig. 4). For example, after w.1 has been computed, both w.2 and v.1 can be computed at the same time. Following the data flow, the increasing subscripts of

$\omega, \dot{\omega}, \dot{v}, \bar{a}, N, F$

show that these variables belong to the set of forward eqations and should be computed forwardly serially, while the variables, f, n, tau should be computed backwardly serially and belong to the set of backward equations. There can be many different assignments of CPUs to computation modules with different set of throughput, computation time and utilization efficiencies.

Assignment One: (see Fig. 5.a)
A simple and pure pipeline with low throughput and long computation time.

Assignment Two: (see Fig. 5.b)
A macro-leveled pipeline with one more local pipeline to increase both computation power and throughput.

Assignment Three: (see Fig. 5.c)
A mesh like special structure. Since Rn, Rv, RN are the most time-consuming stages in the computation process, using more than one process to reducethe computation time can increase the global computation speed and throughput.

The serial nature of the computational data flow lead to the tendency of a pipelined implementation and the parallel nature of the data flow lends itself to multi-processor implementation. Both the seriallism and the parallelism help to increase the computational throughput. However, the characteristics of pipeline and parallel processing still need to be carefully examined to seek for an optimal combination of software algorithm and hardware organization. As to the pipeline, the throughput is mainly effected by the most time-consuming stage in the pipeline; as to parallel processing, the data communication required by software algorithm should take into account both the data dependency of NE equation and the availability of communication channel provided by the hardware.

D. Data Flow Characteristics

A pipeline consists of multiple stages, where each stage performs an operation which is only dependent on data provided by its predecessor stage. This stage in turn provides the output to the successor stage in the pipeline, which can then begin performing their requested operations. Each stage can be performing an operation in parallel with all other stages. Stages need not take the same amount of time, though this case makes the pipeline much harder to synchronize. If a synchronous approach is applied to the pipeline, then the throughput will be dragged down by the slowest stage in the pipeline. If a calculation takes N seconds to perform on a single processor, the same calculation also takes N seconds to perform on the pipeline system while the pipelined system gives increased throughput. However, it will take less than N seconds to perform the same calculation on a mesh-like structure or a macro-level pipeline if the data dependency and hardware structure can support to do so.

The main benefit of a pipeline system is increased computational throughput, as subsequent calculations can be initiated before the result of the current calculation is produced. But a simple pipeline without reducing the longest computation time of the stage in the pipeline is not appropriate since if the computation time of the torque is not less than the sampling time, the robot arm will oscillate. A macro-level pipeline is proposed to further speed up the computation, which assigns more than one processor to one or some stage in the pipeline.

III Transputer-based controller

The objective of this thesis is to utilize a network of transputer, already built on a daughter board of IBM-PC AT, developed by INMOS Inc, to solve the problem of inverse dynamics of the robot arm. Having introduced the dynamic model of a manipulator, a description of the real-world computer component, the programming language, and the integrated computer system are presented as follows to see how these tools implement the computation of the dynamic model.

A. Transputer and Occam language

A transputer is a microcomputer with its own local memory and with hardwire links for connecting one transputer to another transputer by the way of point-to-point communication to support concurrency.

To gain most benefit from the transputer architecture, the whole system can be programmed in OCCAM language. This provides all the advantages of a high level language, the maximum program efficiency and the ability to use the special features of the transputer. OCCAM provides a framework for designing concurrent system using transputers in a direct way. A program running on a transputer is formally equivalent to an OCCAM process, so that a network of transputers can be described directly as an OCCAM program.

On the programming aspect, the software building block is the process. A system is designed in terms of an interconnected set of processes. Each process can be regarded as an independent unit of design. It communicates with other processes along point-to-point channels. Its internal design is transparent, and is specified only by the messages it sends and receives. Communication between processes is self synchronized, removing the need for any separate synchronization mechanism. Internally. each process can be designed as a set of communicating processes. The system design is therefore hierarchically structured. At any level of design, the designer is concerned only with a small and manageable set of processes. Occam is based on these concepts, and provides the definition of the transputer architecture from the logical point of view.

On the communication aspect, to provide synchronized communication, each message must be acknowledged. Consequently, a link required at least one signal wire in each direction. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other by two one-directional signal links, with data transmitted serially. The two signal wires of the link can be used to provide two OCCAM channels, on in each direction. All the transputers will support a standard communications frequency of 10 Mbits/sec regardless of processor performance. Each transputer can provide 4 links(8 one-direction channels) at most.

B. Proposed Architecture

1. Hardware Structure

The hardware facility used to implement the controller of robot manipulator is the IMS B008 TRAM daughterboard. The IMS B008 is a daughterboard which plugs into the IBM PC-AT and has 10 TRAM slots, to accomodate 10 TRAMs (TRAM is board-level transputer that consists of processor, memory, peripheral functions and communication links) When plugged in, they form a pipeline of processing elements which consumes two links for each transputer, while the other two remaining links of each transputer can be " softwired" using an INMOS IMS C004 programmable link switch, incorporated in the the IMS B008.

The transputer network contains at most 10 transputers. The first one is a T800 trasnputer in slot 0 and it is the only one which can communicate with the host computer. That is to say, in order to display the data generated by all the other 9 transputers, data has to send to this monitor transputer in slot 0 and then the monitor transputer calls the I/O function to send the data to the screen. This monitor transputer is also the only one which can directly receive the data from keyboard.

The hard-wired link is represented by a solid line between two transputers while the soft-wired link(or programmed link) is represented by a dash line. The transputer placed in slot 0 is denoted by PM(monitor processor) since it can function like a monitor processor and the P0, P1, ..., P8 are used

to denote the transputers placed in slots1 to 9, all are T800-model except the one in slot 1 which is a T414-model.

2. Algorithms of data flow

Each algorithm is illustrated by an equation table(see Table 1) and two diagrams: linear parallel NE formulation(see Fig. 6) and timing diagram(see Table 2).

IV Experiment Results

The computation time of NE equations using Integer is about 910 micro seconds, while using Real32 the time is about 819 micro seconds. They are less than the 1.5 ms, obtained by Lee[] using pipelined M6800. If only one transputer(T800) is used, the computation time is about 2790 micro seconds. The results of the eight tests are listed below:

For Integer:

Test 1:

computation time is 908 micro seconds

from 4066177 to 4067085

	Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is 3888 with	0	0	0
the tau of joint 2 is 2700 with	0	0	0
the tau of joint 3 is 1728 with	0	0	0
the tau of joint 4 is 972 with	0	0	0
the tau of joint 5 is 432 with	0	0	0
the tau of joint 6 is 108 with	0	0	0

Test 2:

Computation time is 909 micro seconds

from 2846324 to 2847233

	Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is 324 with	0	0	0
the tau of joint 2 is -864 with	90	0	0
the tau of joint 3 is -864 with	90	0	0

the tau of joint 5 is -864 with 0 0 0

the tau of joint 6 is 0 with 90 0 0

Test 3:

computation time is 908 micro seconds

from 3157899 to 3158807

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	-143676 with	0	10	0
the tau of joint 2 is	-116064 with	90	0	0
the tau of joint 3 is	-39264 with	0	10	0
the tau of joint 4 is	37536 with	90	0	0
the tau of joint 5 is	23676 with	0	10	0
the tau of joint 6 is	19200 with	90	0	0

Test 4:

computation time is 909 micro seconds

from 2906363 to 2907272

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	-127356 with	0	10	0
the tau of joint 2 is	-92064 with	90	0	20
the tau of joint 3 is	-21344 with	0	10	0
the tau of joint 4 is	50336 with	90	0	20
the tau of joint 5 is	29116 with	0	10	0
the tau of joint 6 is	19200 with	90	0	20

For Real32:

Test 1:

computation time is 817 micro seconds

from 6192471 to 6193288

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	176.39 with	0	0.0	0.0
the tau of joint 2 is	122.50 with	0	0.0	0.0
the tau of joint 3 is	78.4 with	0	0.0	0.0
the tau of joint 4 is	44.1 with	0	0.0	0.0
the tau of joint 5 is	19.6 with	0	0.0	0.0
the tau of joint 6 is	4.9 with	0	0.0	0.0

Test 2:

computation time is 817 micro seconds

from 4288094 to 4288911

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	14.7 with	0	0.0	0.0
the tau of joint 2 is	-39.2 with	90	0.0	0.0
the tau of joint 3 is	-39.2 with	0	0.0	0.0
the tau of joint 4 is	-39.2 with	90	0.0	0.0
the tau of joint 5 is	-14.7 with	0	0.0	0.0
the tau of joint 6 is	0.0 with	90	0.0	0.0

Test 3:

computation time is 823 micro seconds

from 3831878 to 3832701

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	-3235.30 with	0	0.0	0.0

the tau of joint 2 is -2939.19 with 0 0.0 0.0

the tau of joint 3 is -1339.19 with 0 0.0 0.0

the tau of joint 4 is 260.80 with 0 0.0 0.0

the tau of joint 5 is 235.30 with 0 0.0 0.0

the tau of joint 6 is 400.0 with 0 0.0 0.0

Test 4:

computation time is 823 micro seconds

from 5141659 to 5142482

		Θ	$\dot{\Theta}$	$\ddot{\Theta}$
the tau of joint 1 is	-2895.30 with	0	10.0	0.0
the tau of joint 2 is	-2439.19 with	90	0.0	20.0
the tau of joint 3 is	-965.86 with	0	10.0	0.0
the tau of joint 4 is	527.46 with	90	0.0	20.0
the tau of joint 5 is	348.63 with	0	10.0	0.0
the tau of joint 6 is	400.0 with	90	0.0	20.0

V. Conclusion

The most interesting point is that the floating point computation time is shorter than the fixed point computation time. The T800 transputer contains both a RISC CPU and a Floating Point Unit(FPU). When a floating point operation is in processing, both CPU and FPU are working concurrently, CPU working on address calculation while FPU working on data computation. Therefore, for a simple high-level-language statement, like x-x+1, the computation time with x being integer is still shorter than the computation time with x being floating point. However, for a complex statement, like

```
for k-0 to 7
begin
x[k]:-u[k]+(((r*(z[k]+(r*y[k])))+
      (t*((u[k+3]+(r*[u[k+2]+(r*u[k+1]))))))))+
      (t*((u[k+6]+(r*u[k+5]+(r*u[k+4]))))))))
end.
```

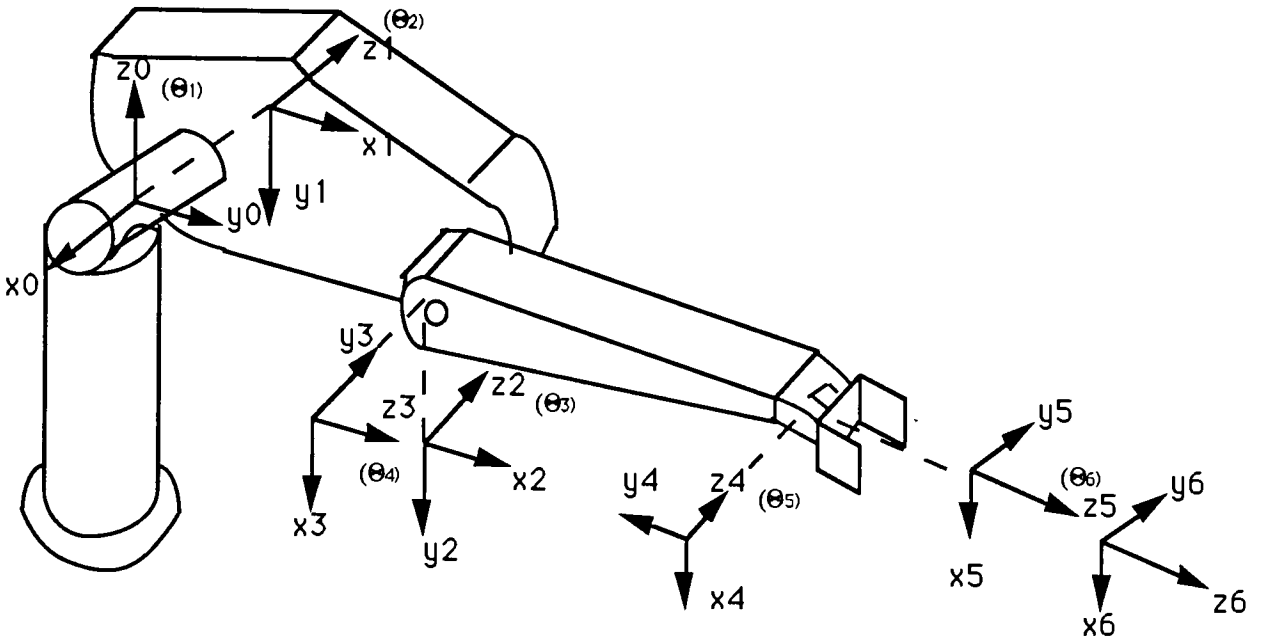
which is a Benchmark, Livermore Loop 7, the computation with data being integer is 178 micro seconds while the computation time with data being floating point is just 85 micro seconds. This is the result of parallel processing when the data type is floating point. When the data type is integer, CPU has to deal with the address and data computation sequentially, thereby slowing down the overall computation.

Besides, there is a scaling problem in fixed point computation. In order to

avoid the difficult of scaling in multiplying six sinusoidal coefficients together, sin and cos values of 0's and 1's only are used in the fixed point computation.

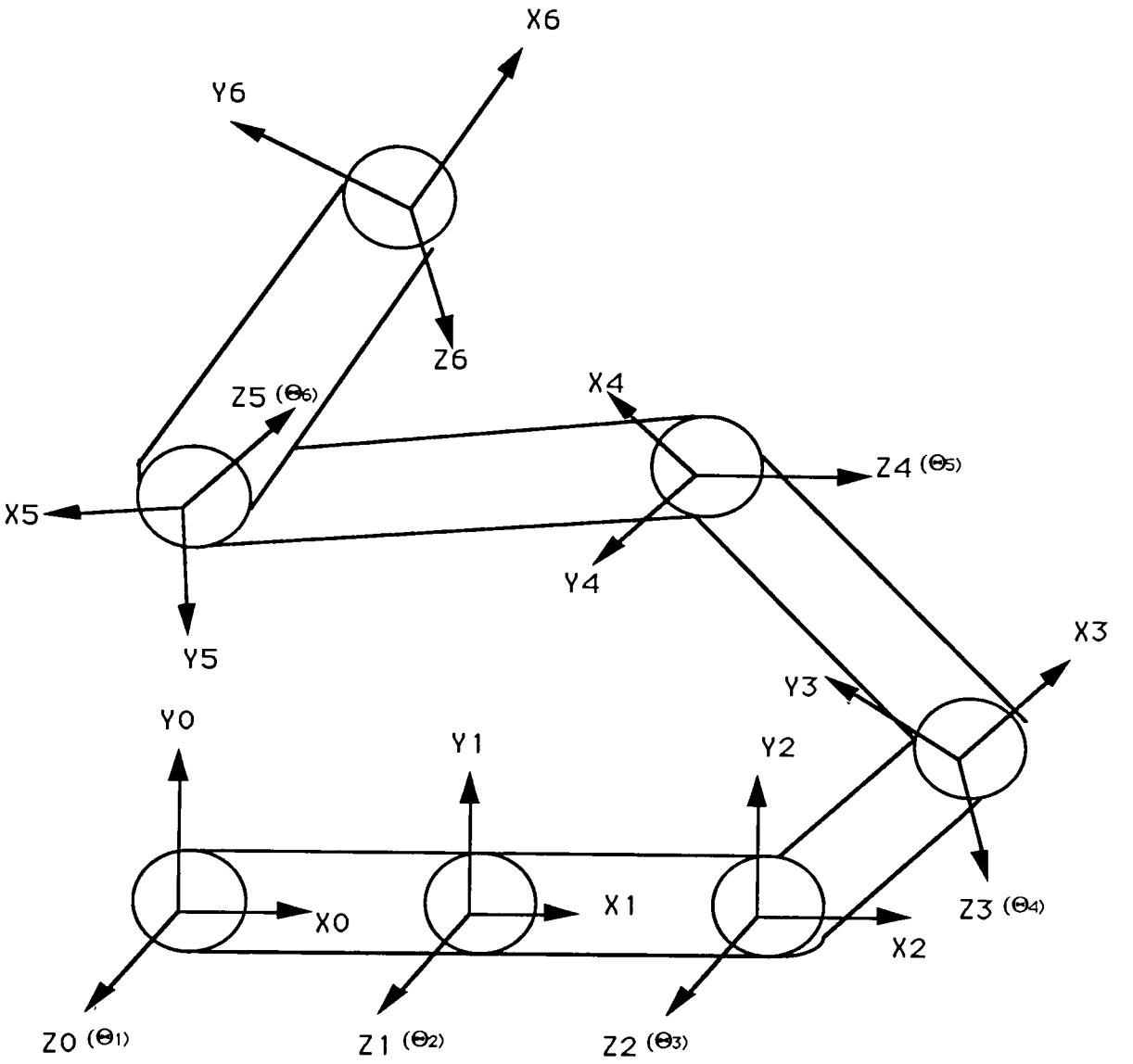
Transputers with OCCAM programs render the real-time computation of robot manipulator control not only possible but also cost-effective. It takes only 0.82 ms to compute the nominal torques of all the six joints of the robot arm by using the parallel recursive algorithm for solving the NE equations of motion. The result obtained by Lee[19] using M68020 is 1.5 ms which is almost twice of that obtained by using transputer network. If a global bus is added to our transputer network to support parallel communication, the overall computation time can be further reduced because of the saving in communication time.

VI Figures and Tables



PUMA ROBOT ARM LINK COORDINATE PARAMETERS					
joint	Θ_i	α_i	a_i	d_i	joint range
1	90	-90	0	0	-160 to +160
2	0	0	431.8 mm	149.09 mm	-225 to 45
3	90	90	-20.32 mm	0	-45 to 225
4	0	-90	0	433.07 mm	-110 to 170
5	0	90	0	0	-100 to 100
6	0	0	0	56.25 mm	-266 to 266

Fig. 1 Establishing link cordinate systems for a PUMA robot



Simple Robot arm link coordinate parameters					
Joint i	Θ_i	α_i	a_i	d_i	Joint range
1	0	0	2 m	0	-180 to 180
2	0	0	2 m	0	-180 to 180
3	0	0	2 m	0	-180 to 180
4	0	0	2 m	0	-180 to 180
5	0	0	2 m	0	-180 to 180
6	0	0	2 m	0	-180 to 180

Fig. 2 Establishing link coordinate systems for a simple robot

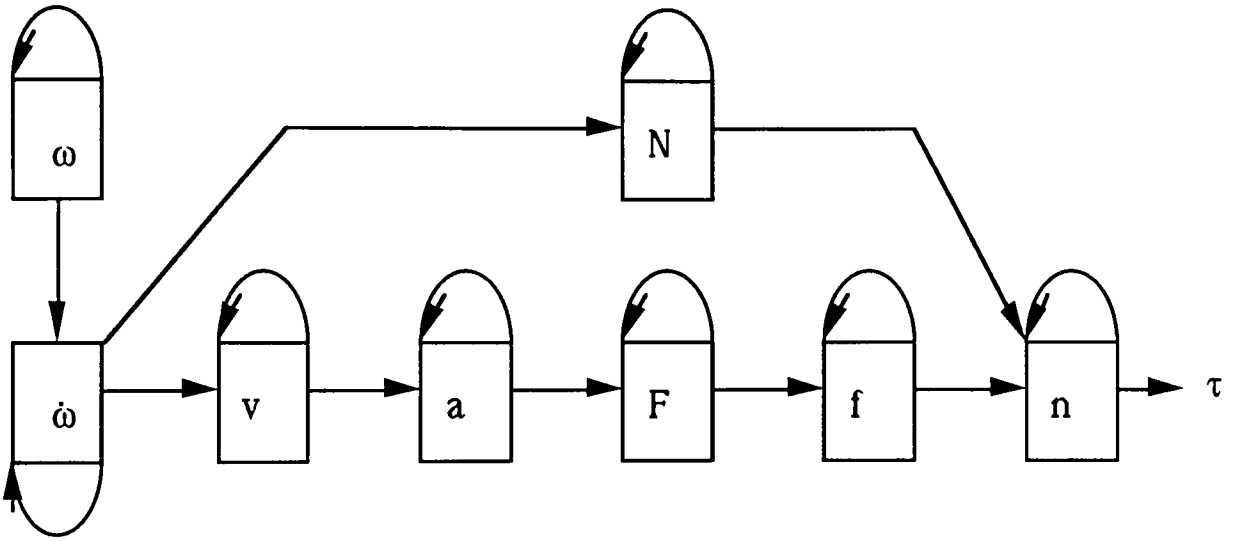


Fig. 3 Serial computation of NE equations

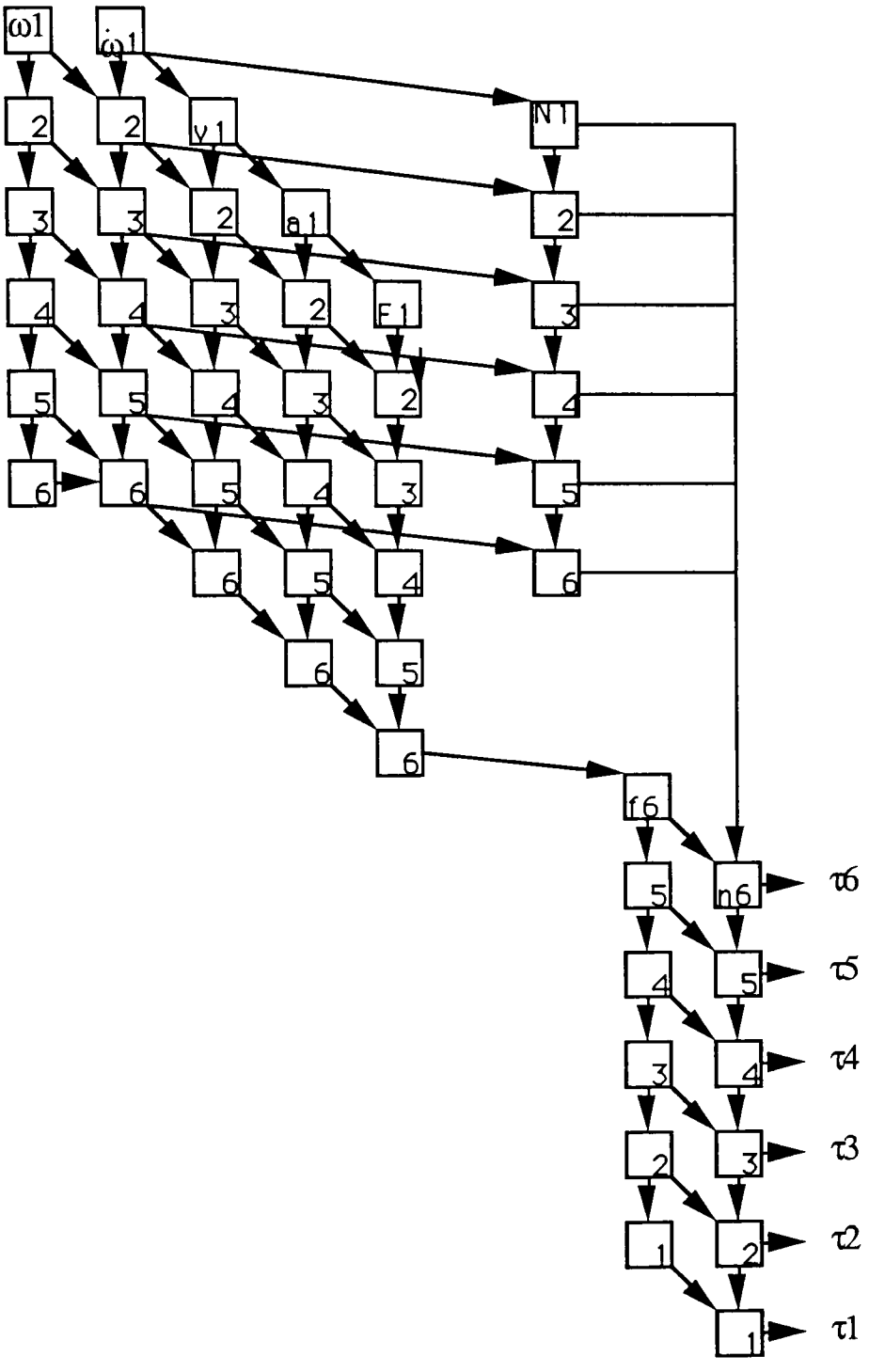


Fig. 4 Linear Parallel NE formulation

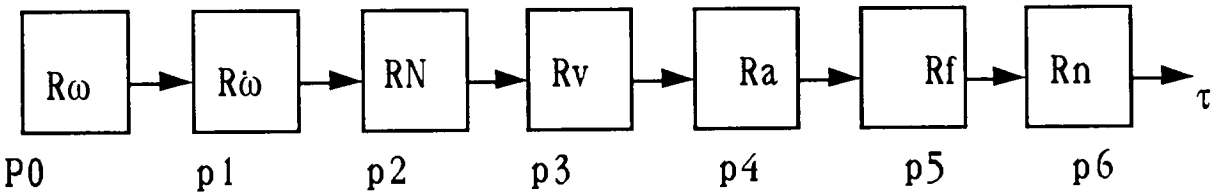


Fig. 5.a Simple pipeline

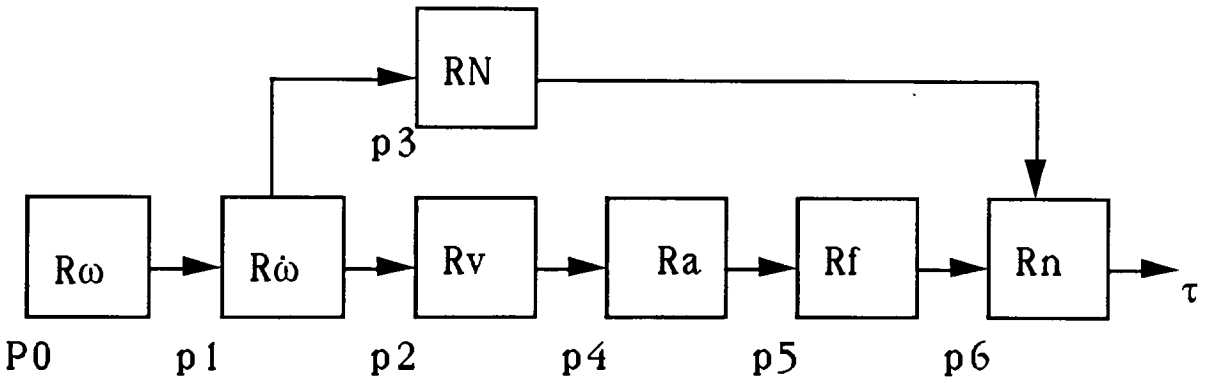


Fig. 5.b Macro-leveled pipeline

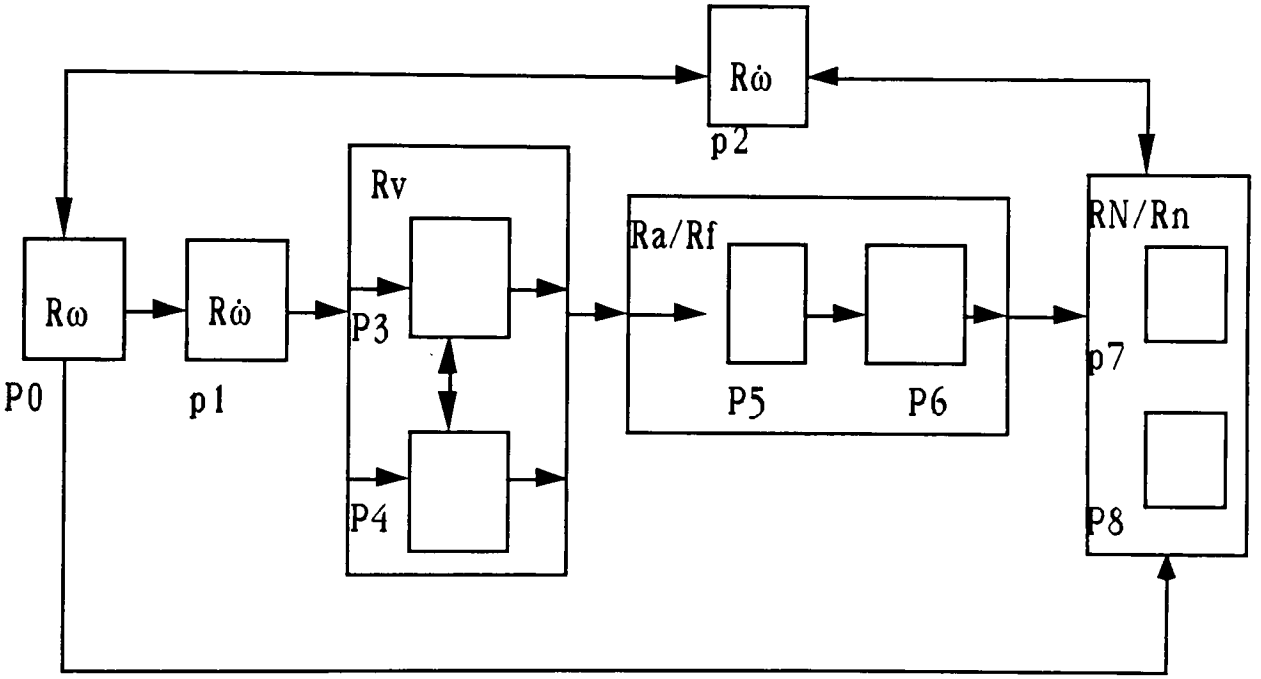


Fig. 5.c Mesh-like pipeline

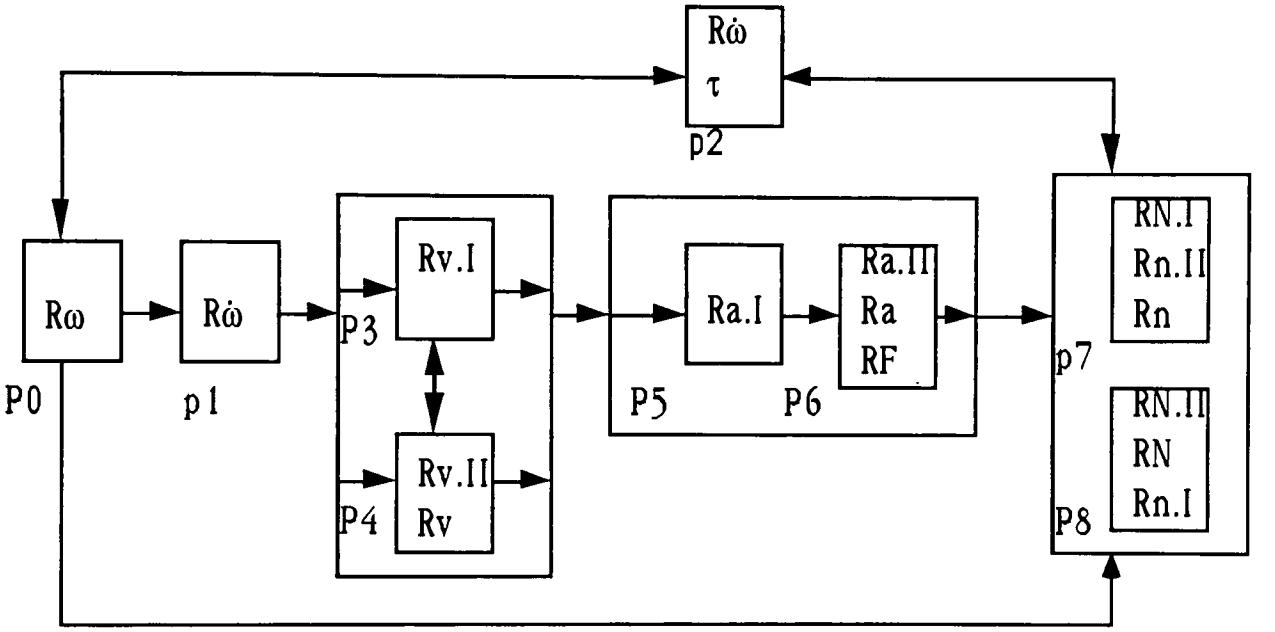


Fig. 6 Configuration of Transputer Network

Table 1: Decomposition of NE equations

$$R_0^i \omega_i = R_{i-1}^i (R_0^{i-1} \omega_{i-1} + z_0 \dot{q}_i)$$

$$R_0^i \dot{\omega}_i = R_{i-1}^i [R_0^{i-1} \dot{\omega}_{i-1} + z_0 \ddot{q}_i + (R_0^{i-1} \omega_{i-1}) \times z_0 \dot{q}_i]$$

$$R_0^i \dot{v}_i = R_0^i \dot{v}_i . I + R_0^i \dot{v}_i . II$$

$$R_0^i \dot{v}_i . I = (R_0^i \dot{\omega}) \times (R_0^i p_i^*)$$

$$R_0^i \dot{v}_i . II = (R_0^i \omega) \times [(R_0^i \omega) \times (R_0^i p_i^*)]$$

$$R_0^i \bar{a}_i = R_0^i \bar{a}_i . I + R_0^i \bar{a}_i . II$$

$$R_0^i \bar{a}_i . I = (R_0^i \dot{\omega}_i) \times (R_0^i \bar{s}_i) + R_0^i \dot{v}_i$$

$$R_0^i \bar{a}_i . II = (R_0^i \omega_i) \times [(R_0^i \omega_i) \times (R_0^i \bar{s}_i)]$$

$$R_0^i F_i = m_i \cdot R_0^i a_i$$

$$R_0^i N_i = R_0^i N_i . I + R_0^i N_i . II$$

$$R_0^i N_i . I = (R_0^i I_i R_i^0)(R_0^i \dot{\omega}_i)$$

$$R_0^i N_i . II = (R_0^i \omega_i) \times [(R_0^i I_i R_i^0) \times (R_0^i \omega_i)]$$

$$R_0^i f_i = R_{i+1}^i (R_0^{i+1} f_{i+1}) + R_0^i F_i$$

$$R_0^i n_i = R_0^i n_i . I + R_0^i n_i . II$$

$$R_0^i n_i . I = R_{i+1}^i [R_0^{i+1} n_{i+1} + (R_0^{i+1} p_i^*) \times (R_0^{i+1} f_{i+1})]$$

$$R_0^i n_i . II = (R_0^i p_i^* + R_0^i \bar{s}_i) \times (R_0^i F_i) + R_0^i N_i$$

$$\tau_i = (R_0^i n_i)^T (R_{i-1}^i z_0) + b_i q_i$$

Table 2: Timing Diagram

	A	B	C	D	E	F	G	H
1	Processor							
2	P0	Rw1	Rw2	Rw3	Rw4	Rw5	Rw6	
3	P1	RW1	RW2	RW3	Rw4	RW5	Rw6	
4	P2	Rw1	Rw2	Rw3	Rw4	Rw5	Rw6	
5	P3		Rv1.I	Rv2.II,Rv2	Rv3.I	Rv4.II,Rv4	Rv5.I	Rv6.II, Rv6
6	P4		Rv1.II,Rv1	Rv2.I	Rv3.II,Rv3	Rv4.I	Rv5.II,Rv5	Rv6.I
7	P5			Ra1.I	Ra2.I	Ra3.I	Ra4.I	Ra5.I
8	P6			Ra1.II, Ra1	Ra2.II, Ra2	Ra3.II, Ra3	Ra4.II, Ra4	Ra5.II, Ra5
9	P7	RN1.I	RN2.I	RN3.I	RN4.I	RN5.I	RN6.I	
10	P8	RN1.II, RN1	RN2.II, RN2	RN3.II, RN3	RN4.II, RN4	RN5.II, RN5	RN6.II, RN6	

Table 2: Timing Diagram

	I	J	K	L	M	N	O	P
1	Table 2: Timing Diagram							
2				Tau 6	Tau 5	Tau 4	Tau 3	Tau 2
3								
4								
5								
6								
7	Ra6.I							
8	Ra6.II, Ra6	RF6, Rf6	RF5, Rf5					
9			Rn6.I, Rn6	Rn5.I, Rn5	Rn4.I, Rn4	Rn3.I, Rn3	Rn2.I, Rn2	Rn1.I, Rn1
10			Rn6,II	Rn5.II	Rn4.II	Rn3.II	Rn2.II	Rn1.II

Table 2: Timing Diagram

	Q
1	
2	Tau 1
3	
4	
5	
6	
7	
8	
9	
10	

VII References

[1] J. Y. S. Luh, and C. S. Lin, "Scheduling of Parallel Computation for a Computer Controlled Mechanical Manipulator," IEEE Trans. Syst. Man, and Cyber., vol. SMC-12, March/April 1982, pp. 214-234.

[2] D. E. Orin, "Pipelined approach to inverse plant plus Jacobian control of robot manipulators," in Proc. 1984 IEEE International conf. Robotics Automation, 1984, pp. 169-175.

[3] A. K. Bejczy, "Robot arm dynamics and control," tech. memo ,33-669. Jet Propulsion Laboratory, Feb. 1974.

[4] C. S. G. Lee, "Robot arm kinematics, dynamics, and control," IEEE Computer, vol. 15, no. 12, pp. 62-80, Dec. 1982.

[5] C. S. G. Lee, Robot Manipulators: Mathematics, Control, and Programming. Cambridge, MA: MIT, 1981.

[6] J. Y. S. Luh, M. W. Walker, and R. P. C. Paul, "On-line computational scheme for mechanical manipulators," Trans. ASME, J. Dynam, Syst., Meas. Contr., vol. 120, pp. 69-76, June 1980.

[7] J. J. Uicker, "On the Dynamic Analysis of Spatial Linkages using 4 x 4 Matrices," Ph.D. dissertation , Northwestern University, Evanston, Ill, 1965.

[8] J. M. Hollerbach, " A Recursive Lagrangian Formulation of Manipulator Dynamics and a Comparative Study of Dynamics Formulation Complexity," IEEE Trans. Systems, Man, Cybern., vol. SMC-10, no. 11, pp. 730-736.

[9] C. S. G., B. H. Lee and R. Nigam, "Development of the Generalized d'Alembert Equations of Motion for Mechanical Manipulatros," Proc.

22nd Conf. Decision and Control, San Antonio, Tex., pp. 1205-1210.

[10] W. M. Armstrong, "Recursive Solution tothe Equations of Motion fo an N-link Manipulator," Proc. 5th World Congr., Thory of Machines, Mechanisms, vol. 2, pp. 1343-1346.

[11] D. E. Orin, R. B. McGhee, M. Vukobratovic and G. Hartoch, "Kinematic and Kinetic Analysis of Open-Chain Linkages Utilizing Newton-Euler Methods," Math. Biosci., vol. 43, pp. 107-130.

[12] J. Denavit and R. S. Hartenberg, " AKinematic Notation for Lower-Pair Mechanisms Based on Matrices," J. App. Mech. vol. 77, pp. 215-221.

[13] IMS B008 User Guide and Reference Manual, INMOS Ltd, Bristol, UK.

[14] The Transputer Databook, INMOS Ltd, Bristol, UK.

[15] Dick Pountain, David May,"A Tutorial Introduction to OCCAM Programming Language", BSP Professional Books, INMOS Ltd, Bristol, UK.

[16] R. D. Dowsing, "Introduction to Concurrency Using OCCAM", VNR Co. Ltd.

[17] J. J. Craig, "Introduction to Robotics, Mechanics and Control", Addison-Wesley Publishing Co.

[18] C. S. G. Lee, R. C. Gonzalez, K. S. Fu, "Tutorial on Robotics", IEEE Computer Society Press.

[19] R. Nigam, C. S. G. Lee, "A Multiprocessor-Based Controller for the Control of Mechanical Manipulators", IEEE Journal of Robotics and Automation, Vol. RA-1, No. 4, Dec. 1985.

VIII Appendices

A. Table of NE Equations of Motion:

Forward equations: i= 1,2,...,n

$$R^i_0\dot{\omega}_i = R^i_{i-1}[R^{i-1}_0\dot{\omega}_{i-1} + z_0\ddot{q}_i + (R^{i-1}_0\omega_{i-1}) \times z_0\dot{q}_i]$$

$$R^i_0\dot{v}_i = (R^i_0\dot{\omega}_i) \times (R^i_0p^*_i) + (R^i_0\omega_i) \times [(R^i_0\omega_i) \times (R^i_0p^*_i)] + R^i_{i-1}(R^{i-1}_0\dot{v}_{i-1})$$

$$R^i_0\bar{a}_i = (R^i_0\dot{\omega}_i) \times (R^i_0\bar{s}_i) + (R^i_0\omega_i) \times [(R^i_0\omega_i) \times (R^i_0\bar{s}_i)] + R^i_0\dot{v}_i$$

$$R^i_0N_i = (R^i_0I_iR^0_i)(R^i_0\dot{\omega}_i) + (R^i_0\omega_i) \times [(R^i_0I_iR^0_i) \times (R^i_0\omega_i)]$$

$$R^i_0F_i = m_iR^i_0\bar{a}_i$$

Backward equations: i=n, n-1,...,1

$$R^i_0f_i = R^i_{i+1}(R^{i+1}_0f_{i+1}) + R^i_0F_i$$

$$R^i_0n_i = R^i_{i+1}[R^{i+1}_0n_{i+1} + (R^{i+1}_0p^*_i) \times (R^{i+1}_0f_{i+1})] + (R^i_0p^*_i + R^i_0\bar{s}_i) \times (R^i_0F_i) + R^i_0N_i$$

$$\tau_i = (R^i_0n_i)^T(R^i_{i-1}z_0) + b_iq_i$$

B. Source Code:

1. Library File of Network File and Monitor File.

```
VAL L0O IS 0:
VAL L1O IS 1:
VAL L2O IS 2:
VAL L3O IS 3:
VAL L0I IS 4:
VAL L1I IS 5:
VAL L2I IS 6:
VAL L3I IS 7:
VAL T.SIN IS [ 0.00(REAL32),  0.0174(REAL32), 0.0348(REAL32),
          0.0523(REAL32), 0.0697(REAL32),
          0.0871(REAL32), 0.104(REAL32),  0.121(REAL32),
          0.139(REAL32),  0.156(REAL32), -- FROM 0 TO 9
          0.173(REAL32),  0.190(REAL32),  0.207(REAL32),
          0.224(REAL32),  0.241(REAL32),
          0.258(REAL32),  0.275(REAL32),  0.292(REAL32),
          0.309(REAL32),  0.325(REAL32), -- FROM 10 TO 19
          0.342(REAL32),  0.358(REAL32),  0.374(REAL32),
          0.390(REAL32),  0.406(REAL32),
          0.422(REAL32),  0.438(REAL32),  0.453(REAL32),
          0.469(REAL32),  0.484(REAL32), -- FROM 20 TO 29
          0.499(REAL32),  0.515(REAL32),  0.529(REAL32),
          0.544(REAL32),  0.559(REAL32),
          0.573(REAL32),  0.587(REAL32),  0.601(REAL32),
          0.615(REAL32),  0.629(REAL32), -- FROM 30 TO 39
          0.642(REAL32),  0.656(REAL32),  0.669(REAL32),
          0.681(REAL32),  0.694(REAL32),
          0.707(REAL32),  0.719(REAL32),  0.731(REAL32),
          0.743(REAL32),  0.754(REAL32), -- FORM 40 TO 49
          0.766(REAL32),  0.777(REAL32),  0.788(REAL32),
          0.798(REAL32),  0.809(REAL32),
          0.819(REAL32),  0.829(REAL32),  0.838(REAL32),
          0.848(REAL32),  0.857(REAL32), -- FROM 50 TO 59
          0.866(REAL32),  0.874(REAL32),  0.882(REAL32),
```

```
          0.891(REAL32), 0.898(REAL32),
          0.906(REAL32), 0.913(REAL32), 0.920(REAL32),
          0.927(REAL32), 0.933(REAL32), -- 60 TO 69
          0.939(REAL32), 0.945(REAL32), 0.951(REAL32),
          0.956(REAL32), 0.961(REAL32),
          0.965(REAL32), 0.970(REAL32), 0.974(REAL32),
          0.978(REAL32), 0.981(REAL32), -- 70 TO 79
          0.984(REAL32), 0.987(REAL32), 0.990(REAL32),
          0.992(REAL32), 0.994(REAL32),
          0.996(REAL32), 0.997(REAL32), 0.998(REAL32),
          0.999(REAL32), 0.999(REAL32), -- 80 TO 89
          1.0(REAL32),  0.999(REAL32), 0.999(REAL32),
          0.998(REAL32), 0.997(REAL32),
          0.996(REAL32), 0.994(REAL32), 0.992(REAL32),
          0.990(REAL32), 0.987(REAL32)]: -- 90 TO 99
VAL T.COS IS [ 1.000(REAL32), 0.999(REAL32), 0.999(REAL32),
          0.998(REAL32), 0.997(REAL32),
          0.996(REAL32), 0.994(REAL32), 0.992(REAL32),
          0.990(REAL32), 0.987(REAL32), -- 0 TO 9
          0.984(REAL32), 0.981(REAL32), 0.978(REAL32),
          0.974(REAL32), 0.970(REAL32),
          0.965(REAL32), 0.961(REAL32), 0.956(REAL32),
          0.951(REAL32), 0.945(REAL32), -- 10 TO 19
          0.939(REAL32), 0.933(REAL32), 0.927(REAL32),
          0.920(REAL32), 0.913(REAL32),
          0.906(REAL32), 0.898(REAL32), 0.891(REAL32),
          0.882(REAL32), 0.874(REAL32), -- 20 TO 29
          0.866(REAL32), 0.857(REAL32), 0.848(REAL32),
          0.838(REAL32), 0.829(REAL32),
          0.819(REAL32), 0.809(REAL32), 0.798(REAL32),
          0.788(REAL32), 0.777(REAL32), -- 30 TO 39
          0.766(REAL32), 0.754(REAL32), 0.743(REAL32),
          0.731(REAL32), 0.719(REAL32),
          0.707(REAL32), 0.694(REAL32), 0.681(REAL32),
          0.669(REAL32), 0.656(REAL32), -- 40 TO 49
          0.642(REAL32), 0.629(REAL32), 0.615(REAL32),
          0.601(REAL32), 0.587(REAL32),
          0.573(REAL32), 0.549(REAL32), 0.544(REAL32),
          0.529(REAL32), 0.515(REAL32), -- 50 TO 59
```

```
          0.500(REAL32), 0.484(REAL32), 0.469(REAL32),
          0.453(REAL32), 0.438(REAL32),
          0.422(REAL32), 0.406(REAL32), 0.390(REAL32),
          0.374(REAL32), 0.358(REAL32), -- 60 TO 69
          0.342(REAL32), 0.325(REAL32), 0.309(REAL32),
          0.292(REAL32), 0.275(REAL32),
          0.258(REAL32), 0.241(REAL32), 0.224(REAL32),
          0.207(REAL32), 0.190(REAL32), -- 70 TO 79
          0.173(REAL32), 0.156(REAL32), 0.139(REAL32),
          0.121(REAL32), 0.104(REAL32),
          0.0871(REAL32), 0.0697(REAL32), 0.0523(REAL32),
          0.0349(REAL32), 0.0174(REAL32), -- 80 TO 89
          0.00(REAL32), -0.0174(REAL32),-0.0348(REAL32),
         -0.0523(REAL32),-0.0697(REAL32),
         -0.0871(REAL32),-0.104(REAL32), -0.121(REAL32),
         -0.139(REAL32), -0.156(REAL32)]: -- 90 TO 99
VAL INT DEG IS 6(INT):
VAL REAL32    ZERO IS  0.0(REAL32):
VAL REAL32     ONE IS  1.0(REAL32):
VAL REAL32   M.ONE IS -1.0(REAL32):
VAL REAL32       G IS 9.8(REAL32):
VAL REAL32     TWO IS 2.0(REAL32):
VAL REAL32     TWE IS 12.0(REAL32):
VAL REAL32       Z IS ZERO:
VAL INT Z.I IS 0(INT):
PROTOCOL ME1I2R IS INT; REAL32; REAL32:
PROTOCOL ME3R IS                REAL32;REAL32;REAL32:
PROTOCOL ME1I3R IS            INT;REAL32;REAL32;REAL32:
PROTOCOL ME1I5R IS  INT;REAL32;REAL32;REAL32;REAL32;REAL32:
PROTOCOL ME6R  IS REAL32;REAL32;REAL32;REAL32;REAL32;REAL32:
PROTOCOL ME63R IS [6][3]REAL32:
PROTOCOL METT
  CASE
    t;  [3]INT
    tau; REAL32
:
PROTOCOL MECC IS [6]INT; [6]REAL32; [6]REAL32:
PROTOCOL ME3RTHE
  CASE
```

```
  THE; INT; REAL32; REAL32
  DAT; REAL32; REAL32; REAL32
:
PROTOCOL ME3RTHE0
 CASE
  THE0; INT
  DAT0; REAL32; REAL32; REAL32
:
VAL REAL32 L IS 1.0(REAL32): -- METER
VAL REAL32 M IS 1.0(REAL32): -- KG
VAL REAL32 S IS (-L)/2.0(REAL32):
VAL INT  ARF1 IS Z.I:
VAL REAL32 a1 IS L:
VAL REAL32 d1 IS Z:

VAL INT  ARF2 IS Z.I:
VAL REAL32 a2 IS L:
VAL REAL32 d2 IS Z:

VAL INT  ARF3 IS Z.I:
VAL REAL32 a3 IS L:
VAL REAL32 d3 IS Z:

VAL INT  ARF4 IS Z.I:
VAL REAL32 a4 IS L:
VAL REAL32 d4 IS Z:

VAL INT  ARF5 IS Z.I:
VAL REAL32 a5 IS L:
VAL REAL32 d5 IS Z:

VAL INT  ARF6 IS Z.I:
VAL REAL32 a6 IS L:
VAL REAL32 d6 IS Z:

VAL REAL32 TWE IS 12.0(REAL32):
VAL REAL32 INER IS (M*(L*L))/TWE:
VAL REAL32 INER1 IS INER:
VAL REAL32 INER2 IS INER:
```

```
VAL REAL32 INER3 IS INER:
VAL REAL32 INER4 IS INER:
VAL REAL32 INER5 IS INER:
VAL REAL32 INER6 IS INER:

VAL REAL32 M1 IS M: -- KG
VAL REAL32 M2 IS M: -- KG
VAL REAL32 M3 IS M: -- KG
VAL REAL32 M4 IS M:
VAL REAL32 M5 IS M:
VAL REAL32 M6 IS M:
VAL LM IS [ M, M, M, M, M, M]: -- LINK MASS
VAL LL IS [ L, L, L, L, L, L]: -- LINK LENGTH
-- RB(I,I-1)- [ C, S, Z]
--          [-S, C, Z]
--          [ Z, Z, 1]
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
-- RF(I-1,I)- [ C,-S, Z]
--          [ S, C, Z]
--          [ Z, Z, 1]
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
VAL RP IS [[ L, Z, Z],[ L, Z, Z],[ L, Z, Z],
        [ L, Z, Z],[ L, Z, Z],[ L, Z, Z]]:
VAL RS IS [[ S, Z, Z],[ S, Z, Z],[ S, Z, Z],
        [ S, Z, Z],[ S, Z, Z],[ S, Z, Z]]:
-- FOR I-1 TO 6
VAL RIR IS [[ Z, Z, Z, Z, INER1, Z, Z, Z, INER1],
        [ Z, Z, Z, Z, INER2, Z, Z, Z, INER2],
        [ Z, Z, Z, Z, INER3, Z, Z, Z, INER3],
        [ Z, Z, Z, Z, INER4, Z, Z, Z, INER4],
        [ Z, Z, Z, Z, INER5, Z, Z, Z, INER5],
```

[Z, Z, Z, Z, INER6, Z, Z, Z, INER6]]:

2. Monitor File which is placed at Monitor Processor

```
#USE uservals
#USE interf
#USE userio    -- the #USE is similar to the "include" of C
#USE LIBTH4
CHAN OF MECC  TO.N: -- comm channel sending data to the
transputer-network
CHAN OF METT  FR.N: -- comm channel receiving data coming from network
PLACE TO.N AT  L2O:
PLACE FR.N AT  L2I:

[3]INT        D.IN:          -- TO GET THE TIMER FROM THE FIRST SLAVE
                        -- [0]-START, [1]-END, [2]-LENGTH
INT anyc, I, J, K:      -- an " any key" buffer.
[6]INT THETA0:
[6]REAL32 THETA1, THETA2:
[6]REAL32 TAU:
VAL TIMEOUT IS 1000000:
TIMER CLOCK:
INT TIMENOW:
SEQ
 SEQ
  -- This example is derived from example 2
  -- It takes its input from a file and throws away the echo

  INT j:
  INT input.error:
  SEQ
   CHAN OF INT filekeys:
   CHAN OF INT keyboard IS filekeys: -- channel from simulated keyboard
   CHAN OF ANY echo:
   CHAN OF ANY screen IS echo: -- echo channel with scope local to this
PAR only
   PAR
```

```
--------------------------------------------------------------
SEQ
  keystream.from.file (from.user.filer[2], to.user.filer[2],
                keyboard, 1, input.error)
  -- check input.error when real screen accessible again
--------------------------------------------------------------
scrstream.sink (screen)   -- consume everything echoed
--------------------------------------------------------------
INT kchar:
SEQ
  kchar := 0
  -- x := 0
  INT x:
  SEQ I=0 FOR 6
    -- WHILE (NOTFINITE(x) OR (x <> -1))) AND
    --     (kchar <> ft.terminated)
    SEQ
      write.char(screen, '>')
      read.echo.char (keyboard, screen, kchar)
      IF
        kchar < 0
          SKIP
        kchar = (INT'#')
          INT hexx RETYPES x:
          read.echo.hex.int (keyboard, screen, hexx, kchar)
        TRUE
          read.echo.int (keyboard, screen, x, kchar)
      IF
        kchar = ft.terminated
          SKIP
        TRUE
          SEQ
            IF
              kchar = ft.number.error
                beep (screen)
              TRUE
```

```
          SKIP
        THETA0[I] := x
REAL32 x:
SEQ I=0 FOR 6
 --WHILE (NOTFINITE(x) OR (x <> -1))) AND
 --     (kchar <> ft.terminated)
 SEQ
  write.char(screen, '>')
  read.echo.char (keyboard, screen, kchar)
  IF
   kchar < 0
    SKIP
   kchar = (INT'#')
    INT hexx RETYPES x:
    read.echo.hex.int (keyboard, screen, hexx, kchar)
   TRUE
    read.echo.real32 (keyboard, screen, x, kchar)
  IF
   kchar = ft.terminated
    SKIP
   TRUE
    SEQ
     IF
      kchar = ft.number.error
       beep (screen)
      TRUE
       SKIP
     THETA1[I] := x
REAL32 x:
SEQ I=0 FOR 6
 -- WHILE (NOTFINITE(x) OR (x <> -1))) AND
 --     (kchar <> ft.terminated)
 SEQ
  write.char(screen, '>')
  read.echo.char (keyboard, screen, kchar)
  IF
```

```
          kchar < 0
            SKIP
          kchar - (INT'#')
            INT hexx RETYPES x:
            read.echo.hex.int (keyboard, screen, hexx, kchar)
          TRUE
            read.echo.real32 (keyboard, screen, x, kchar)
        IF
          kchar - ft.terminated
            SKIP
          TRUE
            SEQ
              IF
                kchar - ft.number.error
                  beep (screen)
                TRUE
                  SKIP
              THETA2[I] := x
      newline (screen)
      IF
        (kchar >- 0) OR (kchar - ft.number.error)
          keystream.sink (keyboard) -- consume the rest of the keyboard
file
        TRUE
          SKIP  -- keyboard file has terminated or failed
      write.endstream (screen) -- terminate scrstream.sink
    ------------------------------------------------------------
 write.full.string(screen, " ANY KEY TO START ...")
 read.char(keyboard, anyc)
 newline(screen)
 TO.N ! THETA0; THETA1; THETA2
 CLOCK ? TIMENOW
 ALT
  FR.N ? CASE t; D.IN  -- to receive the time stamps SENT FROM ZERO
    SEQ
     write.full.string(screen, " the time length is ")
```

```
    write.int(screen, D.IN[2],0)
    write.full.string(screen, " micro seconds")
    newline(screen)
    write.full.string(screen, " start from ")
    write.int(screen, D.IN[0],0)
    newline(screen)
    write.full.string(screen, " ends at ")
    write.int(screen, D.IN[1],0)
    newline(screen)

    SEQ I=0 FOR DEG
      FR.N ? CASE tau; TAU[I]  -- to receive the tau(i) i=1,,6
    SEQ I=1 FOR DEG
      SEQ
       write.full.string(screen, " the tau OF JOINT ")
       write.int(screen, I, 0)
       write.full.string(screen, " IS ")
       write.real32(screen, TAU[DEG-I],5,5)
       write.full.string(screen, " with theta ")
       write.int(screen, THETA0[I-1],5)
       write.real32(screen, THETA1[I-1],5,5)
       write.real32(screen, THETA2[I-1],5,5)
       newline(screen)
    write.full.string(screen, " BACK TO TDS2 ? ")
    read.char(keyboard, anyc)
  CLOCK ? AFTER TIMENOW PLUS TIMEOUT
   SEQ
    write.full.string(screen, " SOMETHING WRONG, BACK TO TDS2 ? ")
    read.char(keyboard, anyc)
```

3. Network File containing 9 processes which are placed at 9 transputers, respectfully.

```
#USE LIBTH4
-- EXPLAINATION OF VARIABLE SHORT HAND
-- 1. M.NO : NUMBER OF MULTIPLICATION OPERATION NEEDED IN THIS
PROCESS
-- 2. A.NO : NUMBER OF ADDITION OPERATION NEEDED IN THIS PROCESS
#USE LIBTH4
PROC ZEROO(CHAN OF MECC FR.H, CHAN OF REAL32 FR.1,
       CHAN OF METT TO.H, CHAN OF ME3RTHE TO.1,
       CHAN OF ME3RTHE TO.2, CHAN OF ME3R TO.8)
 [6]INT THETA0:
 [6]REAL32 THETA1, THETA2:
 TIMER CLOCK:
 [3]INT TIME:              -- FOR TIME STAMP AND TIME LENGTH
 REAL32 V.COS, V.SIN:
 INT I, J, K:
 REAL32 W0, W1, W2, B0, B1, B2, C0, C1, C2:
 [6]REAL32 TAU:            -- TO RECEIVE THE END SIGNAL FROM P8
 SEQ
  -- INIT
  W0:-Z
  W1:-Z
  W2:-Z
  -- receive theta from PH
  FR.H ? THETA0; THETA1; THETA2
  SEQ
   PRI PAR
    SEQ
     -- SEND THETA TO P1 P2
     SEQ J-0 FOR DEG
      SEQ
       PAR
        TO.1 ! THE; THETA0[J]; THETA1[J]; THETA2[J]
        TO.2 ! THE; THETA0[J]; THETA1[J]; THETA2[J]
```

```
CLOCK ? TIME[0]
-- MAIN PROCESS
-- WORKING ON W(I)
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE00 IS THETA0[0]:
THE10 IS THETA1[0]:
SEQ
 V.COS:-T.COS[THE00]
 V.SIN:-T.SIN[THE00]
 -- CALCULATION OF W
 B0:-W0
 B1:-W1
 B2:-W2 + THE10
 W0:- (V.COS *B0) + (V.SIN * B1)
 W1:- (V.COS * B1) - (V.SIN * B0)
 W2:- B2
-- SENDING W(I) TO P1, P2
PAR
 TO.1 ! DAT; W0; W1; W2
 TO.2 ! DAT; W0; W1; W2
 TO.8 ! W0; W1; W2
-- WORKING ON W(I)
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE01 IS THETA0[1]:
THE11 IS THETA1[1]:
SEQ
 V.COS:-T.COS[THE01]
 V.SIN:-T.SIN[THE01]
 -- CALCULATION OF W
 B0:-W0
 B1:-W1
```

```
  B2:=W2 + THE11
  W0:= (V.COS *B0) + (V.SIN * B1)
  W1:= (V.COS * B1) - (V.SIN * B0)
  W2:= B2
-- SENDING W(I) TO P1, P2
PAR
  TO.1 ! DAT; W0; W1; W2
  TO.2 ! DAT; W0; W1; W2
  TO.8 ! W0; W1; W2
-- WORKING ON W(I)
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE02 IS THETA0[2]:
THE12 IS THETA1[2]:
SEQ
  V.COS:=T.COS[THE02]
  V.SIN:=T.SIN[THE02]
  -- CALCULATION OF W
  B0:=W0
  B1:=W1
  B2:=W2 + THE12
  W0:= (V.COS *B0) + (V.SIN * B1)
  W1:= (V.COS * B1) - (V.SIN * B0)
  W2:= B2
-- SENDING W(I) TO P1, P2
PAR
  TO.1 ! DAT; W0; W1; W2
  TO.2 ! DAT; W0; W1; W2
  TO.8 ! W0; W1; W2
-- WORKING ON W(I)
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE03 IS THETA0[3]:
```

```
THE13 IS THETA1[3]:
SEQ
 V.COS:=T.COS[THE03]
 V.SIN:=T.SIN[THE03]
 -- CALCULATION OF W
 B0:=W0
 B1:=W1
 B2:=W2 + THE13
 W0:= (V.COS *B0) + (V.SIN * B1)
 W1:= (V.COS * B1) - (V.SIN * B0)
 W2:= B2
-- SENDING W(I) TO P1, P2
PAR
 TO.1 ! DAT; W0; W1; W2
 TO.2 ! DAT; W0; W1; W2
 TO.8 ! W0; W1; W2
-- WORKING ON W(I)
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE04 IS THETA0[4]:
THE14 IS THETA1[4]:
SEQ
 V.COS:=T.COS[THE04]
 V.SIN:=T.SIN[THE04]
 -- CALCULATION OF W
 B0:=W0
 B1:=W1
 B2:=W2 + THE14
 W0:= (V.COS *B0) + (V.SIN * B1)
 W1:= (V.COS * B1) - (V.SIN * B0)
 W2:= B2
-- SENDING W(I) TO P1, P2, P8
PAR
 TO.1 ! DAT; W0; W1; W2
 TO.2 ! DAT; W0; W1; W2
 TO.8 ! W0; W1; W2
```

```
        -- WORKING ON W(I)
        VAL RB02 IS Z:
        VAL RB12 IS Z:
        VAL RB20 IS Z:
        VAL RB21 IS Z:
        VAL RB22 IS ONE:
        THE05 IS THETA0[5]:
        THE15 IS THETA1[5]:
        SEQ
         V.COS:=T.COS[THE05]
         V.SIN:=T.SIN[THE05]
         -- CALCULATION OF W
         B0:=W0
         B1:=W1
         B2:=W2 + THE15
         W0:= (V.COS *B0) + (V.SIN * B1)
         W1:= (V.COS * B1) - (V.SIN * B0)
         W2:= B2
        -- SENDING W(I) TO P8
        TO.8 ! W0; W1; W2

       SEQ I=0 FOR DEG
        FR.1 ? TAU[I]   -- RECEIVE THE TAU FROM P1
       CLOCK ? TIME[1]
     SEQ
       SKIP
    TIME[2]:= TIME[1]-TIME[0]
    TO.H ! t; TIME
    SEQ I=0 FOR DEG
     TO.H ! tau; TAU[I]
:
#USE LIBTH4
PROC FIRST(CHAN OF ME3RTHE FR.0, CHAN OF ME3R FR.2,
       CHAN OF REAL32 FR.7, CHAN OF REAL32 TO.0,
       CHAN OF INT TO.6, CHAN OF ME3RTHE0 TO.7)
 INT ANYC, I, J:
 [6]INT THETA0:
 [6]REAL32 THETA1, THETA2:
 REAL32 W0, W1, W2, n0, n1, n2:
```

```
REAL32 W.0, W.1, W.2, V.COS, V.SIN, B0, B1, B2, C0, C1, C2:
REAL32 TAU:
-- R(i,i-1)*[ 0, 0, 1], i-6..1
VAL R IS [[ Z, Z, ONE],[ Z, Z, ONE],[ Z, Z, ONE],
      [ Z, Z, ONE],[ Z, Z, ONE],[ Z, Z, ONE]]:
SEQ
 W.0 := ZERO
 W.1 := ZERO
 W.2 := ZERO
 W0 := ZERO
 W1 := ZERO
 W2 := ZERO
 SEQ I=0 FOR DEG
  SEQ
   FR.0 ? CASE THE; THETA0[I]; THETA1[I]; THETA2[I]
   TO.6 ! THETA0[I]
   TO.7 ! THE0; THETA0[I]


 -- WORKING ONE W.1,
 VAL RB02 IS Z:
 VAL RB12 IS Z:
 VAL RB20 IS Z:
 VAL RB21 IS Z:
 VAL RB22 IS ONE:
 THE00 IS THETA0[0]:
 THE10 IS THETA1[0]:
 THE20 IS THETA2[0]:
 SEQ
  V.COS:=T.COS[THE00]
  V.SIN:=T.SIN[THE00]
  -- CALCULATION OF W.i
  B0:=W.0+ (W1*THE10)
  B1:=W.1- (W0*THE10)
  B2:=W.2+         THE20

  W.0:= (V.COS  *B0) + (V.SIN*B1)
  W.1:= (V.COS  *B1) - (V.SIN*B0)
  W.2:=                    RB22 *B2
```

```
 TO.7 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.2,
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE01 IS THETA0[1]:
THE11 IS THETA1[1]:
THE21 IS THETA2[1]:
SEQ
 FR.0 ? CASE DAT; W0; W1; W2
 V.COS:=T.COS[THE01]
 V.SIN:=T.SIN[THE01]
 -- CALCULATION OF W.i
 B0:=W.0+ (W1*THE11)
 B1:=W.1- (W0*THE11)
 B2:=W.2+          THE21

 W.0:= (V.COS  *B0) + (V.SIN*B1)
 W.1:= (V.COS  *B1) - (V.SIN*B0)
 W.2:=                      RB22 *B2
 TO.7 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.3
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE02 IS THETA0[2]:
THE12 IS THETA1[2]:
THE22 IS THETA2[2]:
SEQ
 FR.0 ? CASE DAT; W0; W1; W2
 V.COS:=T.COS[THE02]
 V.SIN:=T.SIN[THE02]
 -- CALCULATION OF W.i
 B0:=W.0+ (W1*THE12)
 B1:=W.1- (W0*THE12)
```

```
  B2:-W.2+          THE22

  W.0:- (V.COS   *B0) + (V.SIN*B1)
  W.1:- (V.COS   *B1) - (V.SIN*B0)
  W.2:-                       RB22 *B2
  TO.7 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.4
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE03 IS THETA0[3]:
THE13 IS THETA1[3]:
THE23 IS THETA2[3]:
SEQ
  FR.0 ? CASE DAT; W0; W1; W2
  V.COS:-T.COS[THE03]
  V.SIN:-T.SIN[THE03]
  -- CALCULATION OF W.i
  B0:-W.0+ (W1*THE13)
  B1:-W.1- (W0*THE13)
  B2:-W.2+          THE23

  W.0:- (V.COS   *B0) + (V.SIN*B1)
  W.1:- (V.COS   *B1) - (V.SIN*B0)
  W.2:-                     B2
  TO.7 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.5
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE04 IS THETA0[4]:
THE14 IS THETA1[4]:
THE24 IS THETA2[4]:
SEQ
  FR.0 ? CASE DAT; W0; W1; W2
```

```
    V.COS:-T.COS[THE04]
    V.SIN:-T.SIN[THE04]
    -- CALCULATION OF W.i
    B0:-W.0+ (W1*THE14)
    B1:-W.1- (W0*THE14)
    B2:-W.2+          THE24

    W.0:- (V.COS  *B0) + (V.SIN*B1)
    W.1:- (V.COS  *B1) - (V.SIN*B0)
    W.2:-                  B2
    TO.7 ! DAT0; W.0; W.1; W.2
  -- WORKING ONE W.6
  VAL RB02 IS Z:
  VAL RB12 IS Z:
  VAL RB20 IS Z:
  VAL RB21 IS Z:
  VAL RB22 IS ONE:
  THE05 IS THETA0[5]:
  THE15 IS THETA1[5]:
  THE25 IS THETA2[5]:
  SEQ
    FR.0 ? CASE DAT; W0; W1; W2
    V.COS:-T.COS[THE05]
    V.SIN:-T.SIN[THE05]
    -- CALCULATION OF W.i
    B0:-W.0+ (W1*THE15)
    B1:-W.1- (W0*THE15)
    B2:-W.2+          THE25

    W.0:- (V.COS  *B0) + (V.SIN*B1)
    W.1:- (V.COS  *B1) - (V.SIN*B0)
    W.2:-                B2
    TO.7 ! DAT0; W.0; W.1; W.2
  SEQ I-0 FOR DEG
    SEQ
      FR.7 ? TAU
      TO.0 ! TAU
:
#USE LIBTH4
```

```
PROC SECON(CHAN OF ME3RTHE FR.0, CHAN OF ME3R TO.1,
      CHAN OF ME3RTHE0 TO.3, CHAN OF ME3RTHE0 TO.4)
 INT I, J,K:
 [6]INT THETA0:
 [6]REAL32 THETA1, THETA2:
 REAL32 W0, W1, W2:
 REAL32 W.0, W.1, W.2:
 REAL32 V.COS, V.SIN, C0, C1, C2, B0, B1, B2:
 SEQ
  W.0:= ZERO
  W.1:= ZERO
  W.2:= ZERO
  W0:= ZERO
  W1:= ZERO
  W2:= ZERO
  SEQ I=0 FOR DEG
   SEQ
    FR.0 ? CASE THE; THETA0[I]; THETA1[I]; THETA2[I]
    PAR
     TO.3 ! THE0; THETA0[I]
     TO.4 ! THE0; THETA0[I]
  -- WORKING ONE W.1,
  VAL RB02 IS Z:
  VAL RB12 IS Z:
  VAL RB20 IS Z:
  VAL RB21 IS Z:
  VAL RB22 IS ONE:
  THE00 IS THETA0[0]:
  THE10 IS THETA1[0]:
  THE20 IS THETA2[0]:
  SEQ
   V.COS:=T.COS[THE00]
   V.SIN:=T.SIN[THE00]
   -- CALCULATION OF W.i
   B0:=W.0+ (W1*THE10)
   B1:=W.1- (W0*THE10)
   B2:=W.2+         THE20

   W.0:= (V.COS  *B0) + (V.SIN*B1)
```

```
 W.1:- (V.COS  *B1) - (V.SIN*B0)
 W.2:-   B2
 PAR
  TO.4 ! DAT0; W0; W1; W2
  TO.3 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.2,
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE01 IS THETA0[1]:
THE11 IS THETA1[1]:
THE21 IS THETA2[1]:
SEQ
 FR.0 ? CASE DAT; W0; W1; W2
 V.COS:-T.COS[THE01]
 V.SIN:-T.SIN[THE01]
 -- CALCULATION OF W.i
 B0:-W.0+ (W1*THE11)
 B1:-W.1- (W0*THE11)
 B2:-W.2+         THE21

 W.0:- (V.COS  *B0) + (V.SIN*B1)
 W.1:- (V.COS  *B1) - (V.SIN*B0)
 W.2:-                   RB22 *B2
 PAR
  TO.3 ! DAT0; W0; W1; W2
  TO.4 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.3
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE02 IS THETA0[2]:
THE12 IS THETA1[2]:
THE22 IS THETA2[2]:
SEQ
```

```
 FR.0 ? CASE DAT; W0; W1; W2
 V.COS:-T.COS[THE02]
 V.SIN:-T.SIN[THE02]
 -- CALCULATION OF W.i
 B0:-W.0+ (W1*THE12)
 B1:-W.1- (W0*THE12)
 B2:-W.2+         THE22

 W.0:- (V.COS  *B0) + (V.SIN*B1)
 W.1:- (V.COS  *B1) - (V.SIN*B0)
 W.2:-                    RB22 *B2
 PAR
   TO.4 ! DAT0; W0; W1; W2
   TO.3 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.4
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE03 IS THETA0[3]:
THE13 IS THETA1[3]:
THE23 IS THETA2[3]:
SEQ
 FR.0 ? CASE DAT; W0; W1; W2
 V.COS:-T.COS[THE03]
 V.SIN:-T.SIN[THE03]
 -- CALCULATION OF W.i
 B0:-W.0+ (W1*THE13)
 B1:-W.1- (W0*THE13)
 B2:-W.2+         THE23

 W.0:- (V.COS  *B0) + (V.SIN*B1)
 W.1:- (V.COS  *B1) - (V.SIN*B0)
 W.2:-                  B2
 PAR
   TO.3 ! DAT0; W0; W1; W2
   TO.4 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.5
```

```
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE04 IS THETA0[4]:
THE14 IS THETA1[4]:
THE24 IS THETA2[4]:
SEQ
  FR.0 ? CASE DAT; W0; W1; W2
  V.COS:-T.COS[THE04]
  V.SIN:-T.SIN[THE04]
  -- CALCULATION OF W.i
  B0:-W.0+ (W1*THE14)
  B1:-W.1- (W0*THE14)
  B2:-W.2+         THE24

  W.0:- (V.COS  *B0) + (V.SIN*B1)
  W.1:- (V.COS  *B1) - (V.SIN*B0)
  W.2:-                B2
  PAR
    TO.4 ! DAT0; W0; W1; W2
    TO.3 ! DAT0; W.0; W.1; W.2
-- WORKING ONE W.6
VAL RB02 IS Z:
VAL RB12 IS Z:
VAL RB20 IS Z:
VAL RB21 IS Z:
VAL RB22 IS ONE:
THE05 IS THETA0[5]:
THE15 IS THETA1[5]:
THE25 IS THETA2[5]:
SEQ
  FR.0 ? CASE DAT; W0; W1; W2
  V.COS:-T.COS[THE05]
  V.SIN:-T.SIN[THE05]
  -- CALCULATION OF W.i
  B0:-W.0+ (W1*THE15)
  B1:-W.1- (W0*THE15)
```

```
    B2:-W.2+          THE25

    W.0:- (V.COS  *B0) + (V.SIN*B1)
    W.1:- (V.COS  *B1) - (V.SIN*B0)
    W.2:-               B2
    PAR
     TO.3 ! DAT0; W0; W1; W2
     TO.4 ! DAT0; W.0; W.1; W.2
:
#USE LIBTH4
PROC THIRD(CHAN OF ME3RTHE0 FR.2, CHAN OF ME3R FR.4,
       CHAN OF ME3R TO.4, CHAN OF ME3R TO.5)
 INT I, J, K:
 [6]INT THETA0:
 REAL32 W0, W1, W2, W.0, W.1, W.2:
 REAL32 V.0, V.1, V.2, V0.I, V1.I, V2.I, V.COS, V.SIN:
 REAL32 C0, C1, C2, B0, B1, B2, V0, V1, V2:
 SEQ
  V.0 := Z
  V.1 := G
  V.2 := Z
  SEQ I=0 FOR DEG
   FR.2 ? CASE THE0; THETA0[I]

  -- WORKING ON V
  -- WORKING ON V1
  THE00 IS THETA0[0]:
  VAL RP0 IS L:
  VAL RP1 IS Z:
  VAL RP2 IS Z:
  SEQ
   FR.2 ? CASE DAT0; W.0; W.1; W.2
   V.COS:=T.COS[THE00]
   V.SIN:=T.SIN[THE00]
   -- CALCULATION OF V.1.II
   -- V(0)=[0,0,G]-[V0.I, V1.I, V2.I], DO R(1,0)V(0)
   -- R(I,I-1)RV.(I-1)
   B0:- (V.COS  *V.0) + (V.SIN*V.1)
   B1:- (V.COS  *V.1) - (V.SIN*V.0)
```

```
 B2:-                    V.2
 -- RW.(I) X RP(I)
 V.0:-((W.1 * RP2) - (W.2 * RP1))+ B0
 V.1:-((W.2 * RP0) - (W.0 * RP2))+ B1
 V.2:-((W.0 * RP1) - (W.1 * RP0))+ B2
 -- TO RECEIVE PARTIAL V.1.I FROM P4
 PAR
  TO.4 ! V.0; V.1; V.2
  TO.5 ! W.0; W.1; W.2
-- WORKING ON V2
THE01 IS THETA0[1]:
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
SEQ  -- WORKING ON V.I 2
 FR.2 ? CASE DAT0; W0; W1; W2

 -- CALCULATION OW(I) X ( RW(I) X RP(I) )
 C0:-(W1 * RP2) - (W2 * RP1)
 C1:-(W2 * RP0) - (W0 * RP2)
 C2:-(W0 * RP1) - (W1 * RP0)
 V.0:-(W1 * C2) - (W2 * C1)
 V.1:-(W2 * C0) - (W0 * C2)
 V.2:-(W0 * C1) - (W1 * C0)
 -- GET PARTIAL RV FROM P4
 FR.4 ? V0; V1; V2
 -- ADD TWO PARTIAL RV
 V.0:- V.0 + V0
 V.1:- V.1 + V1
 V.2:- V.2 + V2
 -- SEND DATA TO P5
 TO.5 ! W0; W1; W2
 TO.5 ! V.0; V.1; V.2
-- WORKING ON V3
THE02 IS THETA0[2]:
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
SEQ
```

```
 FR.2 ? CASE DAT0; W.0; W.1; W.2
 V.COS:=T.COS[THE02]
 V.SIN:=T.SIN[THE02]
 -- CALCULATION OF V.1.II
 -- V(0)=[0,0,G]-[V0.I, V1.I, V2.I], DO R(1,0)V(0)
 -- R(I,I-1)RV.(I-1)
 B0:= (V.COS  *V.0) + (V.SIN*V.1)
 B1:= (V.COS  *V.1) - (V.SIN*V.0)
 B2:=                      V.2
 -- RW.(I) X RP(I)
 V.0:=((W.1 * RP2) - (W.2 * RP1))+ B0
 V.1:=((W.2 * RP0) - (W.0 * RP2))+ B1
 V.2:=((W.0 * RP1) - (W.1 * RP0))+ B2
 -- TO RECEIVE PARTIAL V.1.I FROM P4
 PAR
  TO.4 ! V.0; V.1; V.2
  TO.5 ! W.0; W.1; W.2
-- WORKING ON V4
THE03 IS THETA0[3]:
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
SEQ  -- WORKING ON V.I 4
 FR.2 ? CASE DAT0; W0; W1; W2

 -- CALCULATION OF RW(I) X ( RW(I) X RP(I) )
 C0:=(W1 * RP2) - (W2 * RP1)
 C1:=(W2 * RP0) - (W0 * RP2)
 C2:=(W0 * RP1) - (W1 * RP0)
 V.0:=(W1 * C2) - (W2 * C1)
 V.1:=(W2 * C0) - (W0 * C2)
 V.2:=(W0 * C1) - (W1 * C0)
 -- GET PARTIAL RV FROM P4
 FR.4 ? V0; V1; V2
 -- ADD TWO PARTIAL RV
 V.0:= V.0 + V0
 V.1:= V.1 + V1
 V.2:= V.2 + V2
 -- SEND DATA TO P5
```

```
  TO.5 ! W0; W1; W2
  TO.5 ! V.0; V.1; V.2
-- WORKING ON V5
THE04 IS THETA0[4]:
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
SEQ
  FR.2 ? CASE DAT0; W.0; W.1; W.2
  V.COS:=T.COS[THE04]
  V.SIN:=T.SIN[THE04]
  -- CALCULATION OF V.1.II
  -- V(0)=[0,0,G]-[V0.I, V1.I, V2.I], DO R(1,0)V(0)
  -- R(I,I-1)RV.(I-1)
  B0:= (V.COS  *V.0) + (V.SIN*V.1)
  B1:= (V.COS  *V.1) - (V.SIN*V.0)
  B2:=                  V.2
  -- RW.(I) X RP(I)
  V.0:=((W.1 * RP2) - (W.2 * RP1))+ B0
  V.1:=((W.2 * RP0) - (W.0 * RP2))+ B1
  V.2:=((W.0 * RP1) - (W.1 * RP0))+ B2
  -- TO RECEIVE PARTIAL V.1.I FROM P4
  PAR
    TO.4 ! V.0; V.1; V.2
    TO.5 ! W.0; W.1; W.2
-- WORKING ON V6
THE05 IS THETA0[5]:
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
SEQ  -- WORKING ON V.I 6
  FR.2 ? CASE DAT0; W0; W1; W2
  -- CALCULATION OF RW(I) X ( RW(I) X RP(I) )
  C0:=(W1 * RP2) - (W2 * RP1)
  C1:=(W2 * RP0) - (W0 * RP2)
  C2:=(W0 * RP1) - (W1 * RP0)
  V.0:=(W1 * C2) - (W2 * C1)
  V.1:=(W2 * C0) - (W0 * C2)
  V.2:=(W0 * C1) - (W1 * C0)
```

```
    -- GET PARTIAL RV FROM P4
    FR.4 ? V0; V1; V2
    -- ADD TWO PARTIAL RV
    V.0:= V.0 + V0
    V.1:= V.1 + V1
    V.2:= V.2 + V2
    -- SEND DATA TO P5
    TO.5 ! W0; W1; W2
    TO.5 ! V.0; V.1; V.2
:
#USE LIBTH4
PROC FORTH(CHAN OF ME3RTHE0 FR.2, CHAN OF ME3R FR.3,
        CHAN OF ME3R TO.3, CHAN OF ME3R TO.5)
  INT I,J, K:
  [6]INT THETA0:
  REAL32 V.COS, V.SIN, W0, W1, W2, W.0, W.1, W.2, V.0, V.1, V.2, V0, V1,
V2:
  REAL32 B0, B1, B2, C0, C1, C2:
  SEQ
    -- RECEIVE THETA
    SEQ I=0 FOR DEG
      FR.2 ? CASE THE0; THETA0[I]
    -- WORKING ON PARTIAL RV
    -- WORKING ON RV.1
    VAL RP0 IS L:
    VAL RP1 IS Z:
    VAL RP2 IS Z:
    THE00 IS THETA0[0]:
    SEQ  -- WORKING ON V.I 1
      FR.2 ? CASE DAT0; W0; W1; W2

      -- CALCULATION OF RW(I) X ( RW(I) X RP(I) )
      C0:=(W1 * RP2) - (W2 * RP1)
      C1:=(W2 * RP0) - (W0 * RP2)
      C2:=(W0 * RP1) - (W1 * RP0)
      V.0:=(W1 * C2) - (W2 * C1)
      V.1:=(W2 * C0) - (W0 * C2)
      V.2:=(W0 * C1) - (W1 * C0)
      -- RECEIVE PARTIAL RV FROM P3
```

```
  FR.3 ? V0; V1; V2
  -- ADD TWO PARTIAL RV
  V.0:= V.0 + V0
  V.1:= V.1 + V1
  V.2:= V.2 + V2
  -- SEND DATA TO P5
  TO.5 ! W0; W1; W2
  TO.5 ! V.0; V.1; V.2
-- WORKING ON RV.2
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
THE01 IS THETA0[1]:
SEQ
  FR.2 ? CASE DAT0; W.0; W.1; W.2
  V.COS:=T.COS[THE01]
  V.SIN:=T.SIN[THE01]
  -- DO PARTIAL RV.2
  -- R(I,I-1)RV.(I-1)
  B0:= (V.COS  *V.0) + (V.SIN*V.1)
  B1:= (V.COS  *V.1) - (V.SIN*V.0)
  B2:=  V.2
  -- RW.(I) X RP(I)
  V.0:=((W.1 * RP2) - (W.2 * RP1)) + B0
  V.1:=((W.2 * RP0) - (W.0 * RP2)) + B1
  V.2:=((W.0 * RP1) - (W.1 * RP0)) + B2
  -- SEND PARTIAL RV.2 TO P3
  TO.3 ! V.0; V.1; V.2
  -- SEND DATA TO P5
  TO.5 ! W.0; W.1; W.2
-- WORKING ON RV.3
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
THE02 IS THETA0[2]:
SEQ  -- WORKING ON V.I 3
  FR.2 ? CASE DAT0; W0; W1; W2

  -- CALCULATION OF RW(I) X ( RW(I) X RP(I) )
```

```
 C0:-(W1 * RP2) - (W2 * RP1)
 C1:-(W2 * RP0) - (W0 * RP2)
 C2:-(W0 * RP1) - (W1 * RP0)
 V.0:-(W1 * C2) - (W2 * C1)
 V.1:-(W2 * C0) - (W0 * C2)
 V.2:-(W0 * C1) - (W1 * C0)
 -- RECEIVE PARTIAL RV FROM P3
 FR.3 ? V0; V1; V2
 -- ADD TWO PARTIAL RV
 V.0:- V.0 + V0
 V.1:- V.1 + V1
 V.2:- V.2 + V2
 -- SEND DATA TO P5
 TO.5 ! W0; W1; W2
 TO.5 ! V.0; V.1; V.2
-- WORKING ON RV.4
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
THE03 IS THETA0[3]:
SEQ
 FR.2 ? CASE DAT0; W.0; W.1; W.2
 V.COS:-T.COS[THE03]
 V.SIN:-T.SIN[THE03]
 -- DO PARTIAL RV.2
 -- R(I,I-1)RV.(I-1)
 B0:- (V.COS  *V.0) + (V.SIN*V.1)
 B1:- (V.COS  *V.1) - (V.SIN*V.0)
 B2:- V.2
 -- RW.(I) X RP(I)
 V.0:-((W.1 * RP2) - (W.2 * RP1)) + B0
 V.1:-((W.2 * RP0) - (W.0 * RP2)) + B1
 V.2:-((W.0 * RP1) - (W.1 * RP0)) + B2
 -- SEND PARTIAL RV.2 TO P3
 TO.3 ! V.0; V.1; V.2
 -- SEND DATA TO P5
 TO.5 ! W.0; W.1; W.2
-- WORKING ON RV.5
VAL RP0 IS L:
```

```
VAL RP1 IS Z:
VAL RP2 IS Z:
THE04 IS THETA0[4]:
SEQ  -- WORKING ON V.I 5
 FR.2 ? CASE DAT0; W0; W1; W2

 -- CALCULATION OF RW(I) X ( RW(I) X RP(I) )
 C0:=(W1 * RP2) - (W2 * RP1)
 C1:=(W2 * RP0) - (W0 * RP2)
 C2:=(W0 * RP1) - (W1 * RP0)
 V.0:=(W1 * C2) - (W2 * C1)
 V.1:=(W2 * C0) - (W0 * C2)
 V.2:=(W0 * C1) - (W1 * C0)
 -- RECEIVE PARTIAL RV FROM P3
 FR.3 ? V0; V1; V2
 -- ADD TWO PARTIAL RV
 V.0:= V.0 + V0
 V.1:= V.1 + V1
 V.2:= V.2 + V2
 -- SEND DATA TO P5
 TO.5 ! W0; W1; W2
 TO.5 ! V.0; V.1; V.2
-- WORKING ON RV.6
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
THE05 IS THETA0[5]:
SEQ
 FR.2 ? CASE DAT0; W.0; W.1; W.2
 V.COS:=T.COS[THE05]
 V.SIN:=T.SIN[THE05]
 -- DO PARTIAL RV.2
 -- R(I,I-1)RV.(I-1)
 B0:= (V.COS  *V.0) + (V.SIN*V.1)
 B1:= (V.COS  *V.1) - (V.SIN*V.0)
 B2:= RB22*V.2
 -- RW.(I) X RP(I)
 V.0:=((W.1 * RP2) - (W.2 * RP1)) + B0
 V.1:=((W.2 * RP0) - (W.0 * RP2)) + B1
```

```
    V.2:=((W.0 * RP1) - (W.1 * RP0)) + B2
    -- SEND PARTIAL RV.2 TO P3
    TO.3 ! V.0; V.1; V.2
    -- SEND DATA TO P5
    TO.5 ! W.0; W.1; W.2
:
#USE LIBTH4
PROC FIFTH(CHAN OF ME3R FR.3, CHAN OF ME3R FR.4, CHAN OF ME3R TO.6)
 REAL32 I, J, K:
 REAL32 W0, W1, W2, W.0, W.1, W.2, V0, V1, V2, V0I, V1I, V2I:
 REAL32 B0, B1, B2, C0, C1, C2:
 REAL32 A0, A1, A2:
 SEQ
  VAL RS0 IS S:
  VAL RS1 IS Z:
  VAL RS2 IS Z:
  SEQ  -- WORKING ON A1
   PAR
    SEQ
     FR.4 ? W0; W1; W2
     FR.4 ? V0; V1; V2
    FR.3 ? W.0; W.1; W.2
   TO.6 ! W0; W1; W2
   -- RW.(I) X RS(I)
   C0:=(W1 * RS2)-(W2 * RS1)
   C1:=(W2 * RS0)-(W0 * RS2)
   C2:=(W0 * RS1)-(W1 * RS0)
   A0:=(W1 * C2)-(W2 * C1)
   A1:=(W2 * C0)-(W0 * C2)
   A2:=(W0 * C1)-(W1 * C0)
   A0:=A0+ V0
   A1:=A1+ V1
   A2:=A2+ V2
   TO.6 ! A0; A1; A2
  VAL RS0 IS S:
  VAL RS1 IS Z:
  VAL RS2 IS Z:
  SEQ  -- WORKING ON A2
   PAR
```

```
    SEQ
      FR.3 ? W0; W1; W2
      FR.3 ? V0; V1; V2
    FR.4 ? W.0; W.1; W.2
  TO.6 ! W.0; W.1; W.2
  -- RW.(I) X RS(I)
  C0:=(W1 * RS2)-(W2 * RS1)
  C1:=(W2 * RS0)-(W0 * RS2)
  C2:=(W0 * RS1)-(W1 * RS0)
  A0:=(W1 * C2)-(W2 * C1)
  A1:=(W2 * C0)-(W0 * C2)
  A2:=(W0 * C1)-(W1 * C0)
  A0:=A0+ V0
  A1:=A1+ V1
  A2:=A2+ V2
  TO.6 ! A0; A1; A2
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ  -- WORKING ON A3
  PAR
    SEQ
      FR.4 ? W0; W1; W2
      FR.4 ? V0; V1; V2
    FR.3 ? W.0; W.1; W.2
  TO.6 ! W.0; W.1; W.2
  -- RW.(I) X RS(I)
  C0:=(W1 * RS2)-(W2 * RS1)
  C1:=(W2 * RS0)-(W0 * RS2)
  C2:=(W0 * RS1)-(W1 * RS0)
  A0:=(W1 * C2)-(W2 * C1)
  A1:=(W2 * C0)-(W0 * C2)
  A2:=(W0 * C1)-(W1 * C0)
  A0:=A0+ V0
  A1:=A1+ V1
  A2:=A2+ V2
  TO.6 ! A0; A1; A2
VAL RS0 IS S:
VAL RS1 IS Z:
```

```
VAL RS2 IS Z:
SEQ  -- WORKING ON A4
 PAR
  SEQ
    FR.3 ? W0; W1; W2
    FR.3 ? V0; V1; V2
  FR.4 ? W.0; W.1; W.2
 TO.6 ! W.0; W.1; W.2
 -- RW.(I) X RS(I)
 C0:=(W1 * RS2)-(W2 * RS1)
 C1:=(W2 * RS0)-(W0 * RS2)
 C2:=(W0 * RS1)-(W1 * RS0)
 A0:=(W1 * C2)-(W2 * C1)
 A1:=(W2 * C0)-(W0 * C2)
 A2:=(W0 * C1)-(W1 * C0)
 A0:=A0+ V0
 A1:=A1+ V1
 A2:=A2+ V2
 TO.6 ! A0; A1; A2
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ  -- WORKING ON A5
 PAR
  SEQ
    FR.4 ? W0; W1; W2
    FR.4 ? V0; V1; V2
  FR.3 ? W.0; W.1; W.2
 TO.6 ! W.0; W.1; W.2
 -- RW.(I) X RS(I)
 C0:=(W1 * RS2)-(W2 * RS1)
 C1:=(W2 * RS0)-(W0 * RS2)
 C2:=(W0 * RS1)-(W1 * RS0)
 A0:=(W1 * C2)-(W2 * C1)
 A1:=(W2 * C0)-(W0 * C2)
 A2:=(W0 * C1)-(W1 * C0)
 A0:=A0+ V0
 A1:=A1+ V1
 A2:=A2+ V2
```

```
    TO.6 ! A0; A1; A2
   VAL RS0 IS S:
   VAL RS1 IS Z:
   VAL RS2 IS Z:
   SEQ  -- WORKING ON A6
    PAR
     SEQ
       FR.3 ? W0; W1; W2
       FR.3 ? V0; V1; V2
     FR.4 ? W.0; W.1; W.2
    TO.6 ! W.0; W.1; W.2
    -- RW.(I) X RS(I)
    C0:=(W1 * RS2)-(W2 * RS1)
    C1:=(W2 * RS0)-(W0 * RS2)
    C2:=(W0 * RS1)-(W1 * RS0)
    A0:=(W1 * C2)-(W2 * C1)
    A1:=(W2 * C0)-(W0 * C2)
    A2:=(W0 * C1)-(W1 * C0)
    A0:=A0+ V0
    A1:=A1+ V1
    A2:=A2+ V2
    TO.6 ! A0; A1; A2
:
#USE LIBTH4
PROC SIXTH(CHAN OF INT FR.1, CHAN OF ME3R FR.5, CHAN OF ME3R TO.7,
TO.8)
 -- working on (1) RN.I
 -- WORKING ON (2) f(i) and F(i)

 INT I, J, K:
 INT THETA00, THETA01, THETA02, THETA03, THETA04, THETA05:
 REAL32 A00, A01, A02, A10, A11, A12, A20, A21, A22,
    A30, A31, A32, A40, A41, A42, A50, A51, A52:
 REAL32 A0, A1, A2:
 REAL32 W0, W1, W2, N0, N1, N2, B0, B1, B2, V.COS, V.SIN:
 REAL32 F0, F1, F2, C0, C1, C2:
 REAL32 f0, f1, f2:
 REAL32 W.0, W.1, W.2:
 SEQ
```

```
f0:- ZERO
f1:- ZERO
f2:- ZERO
SEQ
  FR.1 ? THETA00
  FR.1 ? THETA01
  FR.1 ? THETA02
  FR.1 ? THETA03
  FR.1 ? THETA04
  FR.1 ? THETA05
--
SEQ
  VAL RS0 IS S:
  VAL RS1 IS Z:
  VAL RS2 IS Z:
  SEQ
    -- RW(I) X [RW(I) X RS(I)]
    FR.5 ? W.0; W.1; W.2
    -- RW.(I) X RS(I)
    A00:-(W.1 * RS2) - (W.2 * RS1)
    A01:-(W.2 * RS0) - (W.0 * RS2)
    A02:-(W.0 * RS1) - (W.1 * RS0)
    FR.5 ? A0; A1; A2
    A00:- A00 + A0
    A01:- A01 + A1
    A02:- A02 + A2
  VAL RS0 IS S:
  VAL RS1 IS Z:
  VAL RS2 IS Z:
  SEQ
    -- RW(I) X [RW(I) X RS(I)]
    FR.5 ? W.0; W.1; W.2
    -- RW.(I) X RS(I)
    A10:-(W.1 * RS2) - (W.2 * RS1)
    A11:-(W.2 * RS0) - (W.0 * RS2)
    A12:-(W.0 * RS1) - (W.1 * RS0)
    FR.5 ? A0; A1; A2
    A10:- A10 + A0
    A11:- A11 + A1
```

```
 A12:- A12 + A2
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
  -- RW(I) X [RW(I) X RS(I)]
  FR.5 ? W.0; W.1; W.2
  -- RW.(I) X RS(I)
  A20:-(W.1 * RS2) - (W.2 * RS1)
  A21:-(W.2 * RS0) - (W.0 * RS2)
  A22:-(W.0 * RS1) - (W.1 * RS0)
  FR.5 ? A0; A1; A2
  A20:- A20 + A0
  A21:- A21 + A1
  A22:- A22 + A2
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
  -- RW(I) X [RW(I) X RS(I)]
  FR.5 ? W.0; W.1; W.2
  -- RW.(I) X RS(I)
  A30:-(W.1 * RS2) - (W.2 * RS1)
  A31:-(W.2 * RS0) - (W.0 * RS2)
  A32:-(W.0 * RS1) - (W.1 * RS0)
  FR.5 ? A0; A1; A2
  A30:- A30 + A0
  A31:- A31 + A1
  A32:- A32 + A2
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
  -- RW(I) X [RW(I) X RS(I)]
  FR.5 ? W.0; W.1; W.2
  -- RW.(I) X RS(I)
  A40:-(W.1 * RS2) - (W.2 * RS1)
  A41:-(W.2 * RS0) - (W.0 * RS2)
  A42:-(W.0 * RS1) - (W.1 * RS0)
```

```
    FR.5 ? A0; A1; A2
    A40:- A40 + A0
    A41:- A41 + A1
    A42:- A42 + A2
   VAL RS0 IS S:
   VAL RS1 IS Z:
   VAL RS2 IS Z:
   SEQ
    -- RW(I) X [RW(I) X RS(I)]
    FR.5 ? W.0; W.1; W.2
    -- RW.(I) X RS(I)
    A50:-(W.1 * RS2) - (W.2 * RS1)
    A51:-(W.2 * RS0) - (W.0 * RS2)
    A52:-(W.0 * RS1) - (W.1 * RS0)
    FR.5 ? A0; A1; A2
    A50:- A50 + A0
    A51:- A51 + A1
    A52:- A52 + A2
 -- WORKING ON F AND f
 -- WORKING ON F6, f6
 VAL RF02 IS Z:
 VAL RF12 IS Z:
 VAL RF20 IS Z:
 VAL RF21 IS Z:
 VAL RF22 IS ONE:
 SEQ
  F0:-M * A50
  F1:-M * A51
  F2:-M * A52
  PAR
   TO.8 ! F0; F1; F2  -- F6 IS SENT TO P7
   TO.7 ! f0; f1; f2  -- f7 IS SENT TO P8
  -- DO f6
  f0:-f0 + F0    -- ASSUME f(7)-[0, 0, 0]
  f1:-f1 + F1
  f2:-f2 + F2
-- WORKING ON F5, f5
VAL RF02 IS Z:
VAL RF12 IS Z:
```

```
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
 -- DO F5, F4, ..., F1
  F0:= M * A40
  F1:= M * A41
  F2:= M * A42
  PAR
   TO.8 ! F0; F1; F2  -- SENT F5
   TO.7 ! f0; f1; f2  -- SENT f6
 -- DO f5, f4, ..., f1
 V.COS:=T.COS[THETA05]
 V.SIN:=T.SIN[THETA05]
 B0:= (V.COS  *f0) - (V.SIN*f1)
 B1:= (V.SIN  *f0) + (V.COS*f1)
 B2:= RF22*f2
 f0:= B0 + F0
 f1:= B1 + F1
 f2:= B2 + F2
-- WORKING ON F4, f4
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
 -- DO F5, F4, ..., F1
  F0:= M * A30
  F1:= M * A31
  F2:= M * A32
  PAR
   TO.8 ! F0; F1; F2  -- SENT F5
   TO.7 ! f0; f1; f2  -- SENT f6
 -- DO f5, f4, ..., f1
 V.COS:=T.COS[THETA04]
 V.SIN:=T.SIN[THETA04]
 B0:= (V.COS  *f0) - (V.SIN*f1)
 B1:= (V.SIN  *f0) + (V.COS*f1)
```

```
  B2:- RF22*f2
  f0:- B0 + F0
  f1:- B1 + F1
  f2:- B2 + F2
-- WORKING ON F3, f3
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
  -- DO F5, F4, ..., F1
  F0:- M * A20
  F1:- M * A21
  F2:- M * A22
  PAR
    TO.8 ! F0; F1; F2  -- SENT F5
    TO.7 ! f0; f1; f2  -- SENT f6
  -- DO f5, f4, ..., f1
  V.COS:-T.COS[THETA03]
  V.SIN:-T.SIN[THETA03]
  B0:- (V.COS   *f0) - (V.SIN*f1)
  B1:- (V.SIN   *f0) + (V.COS*f1)
  B2:- RF22*f2
  f0:- B0 + F0
  f1:- B1 + F1
  f2:- B2 + F2
-- WORKING ON F2, f2
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
  -- DO F5, F4, ..., F1
  F0:- M * A10
  F1:- M * A11
  F2:- M * A12
  PAR
```

```
      TO.8 ! F0; F1; F2  -- SENT F5
      TO.7 ! f0; f1; f2  -- SENT f6
    -- DO f5, f4, ..., f1
    V.COS:=T.COS[THETA02]
    V.SIN:=T.SIN[THETA02]
    B0:= (V.COS  *f0) - (V.SIN*f1)
    B1:= (V.SIN  *f0) + (V.COS*f1)
    B2:= RF22*f2
    f0:= B0 + F0
    f1:= B1 + F1
    f2:= B2 + F2
  -- WORKING ON F1, f1
  VAL RF02 IS Z:
  VAL RF12 IS Z:
  VAL RF20 IS Z:
  VAL RF21 IS Z:
  VAL RF22 IS ONE:
  SEQ
    -- DO F5, F4, ..., F1
    F0:= M * A00
    F1:= M * A01
    F2:= M * A02
    PAR
      TO.8 ! F0; F1; F2  -- SENT F5
      TO.7 ! f0; f1; f2  -- SENT f6
    -- DO f5, f4, ..., f1
    V.COS:=T.COS[THETA01]
    V.SIN:=T.SIN[THETA01]
    B0:= (V.COS  *f0) - (V.SIN*f1)
    B1:= (V.SIN  *f0) + (V.COS*f1)
    B2:= RF22*f2
    f0:= B0 + F0
    f1:= B1 + F1
    f2:= B2 + F2
:
#USE LIBTH4
PROC SEVEN(CHAN OF ME3RTHE0 FR.1, CHAN OF ME3R FR.6,
           CHAN OF ME3R FR.8, CHAN OF REAL32 TO.1, CHAN OF ME3R TO.8)
 INT I, J, K:
```

```
INT THETA00, THETA01, THETA02, THETA03, THETA04, THETA05:
REAL32 W0, W1, W2, W.0, W.1, W.2, n0, n1, n2, nI0, nI1, nI2:
REAL32 V.COS, V.SIN, C0, C1, C2, B0, B1, B2, f0, f1, f2:
REAL32 N0, N1, N2:
SEQ
 FR.1 ? CASE THE0; THETA00
 FR.1 ? CASE THE0; THETA01
 FR.1 ? CASE THE0; THETA02
 FR.1 ? CASE THE0; THETA03
 FR.1 ? CASE THE0; THETA04
 FR.1 ? CASE THE0; THETA05

 VAL RIR0 IS Z:
 VAL RIR1 IS Z:
 VAL RIR2 IS Z:
 VAL RIR3 IS Z:
 VAL RIR4 IS INER:
 VAL RIR5 IS Z:
 VAL RIR6 IS Z:
 VAL RIR7 IS Z:
 VAL RIR8 IS INER:
 SEQ
  FR.1 ? CASE DAT0; W.0; W.1; W.2
  -- CALCULATION OF RN.II
  -- (RIR)(RW.)
  N0:= (RIR0 * W.0) + (RIR1 * W.1)
  N1:= (RIR3 * W.0) + (RIR4 * W.1)
  N2:= (RIR6 * W.0) + (RIR7 * W.1)
  N0:= N0 +(RIR2 * W.2)
  N1:= N1 +(RIR5 * W.2)
  N2:= N2 +(RIR8 * W.2)
  TO.8 ! N0; N1; N2
 VAL RIR0 IS Z:
 VAL RIR1 IS Z:
 VAL RIR2 IS Z:
 VAL RIR3 IS Z:
 VAL RIR4 IS INER:
 VAL RIR5 IS Z:
 VAL RIR6 IS Z:
```

```
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
 FR.1 ? CASE DAT0; W.0; W.1; W.2
 -- CALCULATION OF RN.II
 -- (RIR)(RW.)
 N0:= (RIR0 * W.0) + (RIR1 * W.1)
 N1:= (RIR3 * W.0) + (RIR4 * W.1)
 N2:= (RIR6 * W.0) + (RIR7 * W.1)
 N0:= N0 +(RIR2 * W.2)
 N1:= N1 +(RIR5 * W.2)
 N2:= N2 +(RIR8 * W.2)
 TO.8 ! N0; N1; N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER:
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
 FR.1 ? CASE DAT0; W.0; W.1; W.2
 -- CALCULATION OF RN.II
 -- (RIR)(RW.)
 N0:= (RIR0 * W.0) + (RIR1 * W.1)
 N1:= (RIR3 * W.0) + (RIR4 * W.1)
 N2:= (RIR6 * W.0) + (RIR7 * W.1)
 N0:= N0 +(RIR2 * W.2)
 N1:= N1 +(RIR5 * W.2)
 N2:= N2 +(RIR8 * W.2)
 TO.8 ! N0; N1; N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER:
VAL RIR5 IS Z:
```

```
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
 FR.1 ? CASE DAT0; W.0; W.1; W.2
 -- CALCULATION OF RN.II
 -- (RIR)(RW.)
 N0:- (RIR0 * W.0) + (RIR1 * W.1)
 N1:- (RIR3 * W.0) + (RIR4 * W.1)
 N2:- (RIR6 * W.0) + (RIR7 * W.1)
 N0:- N0 +(RIR2 * W.2)
 N1:- N1 +(RIR5 * W.2)
 N2:- N2 +(RIR8 * W.2)
 TO.8 ! N0; N1; N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER:
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
 FR.1 ? CASE DAT0; W.0; W.1; W.2
 -- CALCULATION OF RN.II
 -- (RIR)(RW.)
 N0:- (RIR0 * W.0) + (RIR1 * W.1)
 N1:- (RIR3 * W.0) + (RIR4 * W.1)
 N2:- (RIR6 * W.0) + (RIR7 * W.1)
 N0:- N0 +(RIR2 * W.2)
 N1:- N1 +(RIR5 * W.2)
 N2:- N2 +(RIR8 * W.2)
 TO.8 ! N0; N1; N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER:
```

```
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
 FR.1 ? CASE DAT0; W.0; W.1; W.2
 -- CALCULATION OF RN.II
 -- (RIR)(RW.)
 N0:= (RIR0 * W.0) + (RIR1 * W.1)
 N1:= (RIR3 * W.0) + (RIR4 * W.1)
 N2:= (RIR6 * W.0) + (RIR7 * W.1)
 N0:= N0 +(RIR2 * W.2)
 N1:= N1 +(RIR5 * W.2)
 N2:= N2 +(RIR8 * W.2)
 TO.8 ! N0; N1; N2
-- WORKING ON n6
-- n6= N6 + (RP*I + RS(I) X RF(I) WHICH IS DONE BY P8
PAR
 FR.6 ? f0; f1; f2  -- f(7) FROM P6
 FR.8 ? n0; n1; n2  -- n(6) FROM P8
TO.1 ! n2
-- WORKING ON n5
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
 FR.6 ? f0; f1; f2  -- f6 FROM P6
 V.COS:=T.COS[THETA05]
 V.SIN:=T.SIN[THETA05]
 -- CALCULATION OF RP(I+1,I) X RF(I+1)-( R(I+1,I)RP(I)) X RF(I+1)
 B0:= V.COS * L
 B1:= V.SIN * L
 C0:= n0 - (B1*f2)
 C1:= n1 - (B0*f2)
 C2:= (B0*f1)+(B1*f0)
 C2:= C2 + n2
 -- CALCULATION OF R(I, I+1)*( R(I+1, I)n(I+1) + RP(I+1,I) X RF(I+1)
```

```
  B0:= (V.COS * C0) - (V.SIN * C1)
  B1:= (V.SIN * C0) + (V.COS * C1)
  B2:=  C2
  FR.8 ? nI0; nI1; nI2
  n0:= B0 + nI0
  n1:= B1 + nI1
  n2:= B2 + nI2
  TO.1 ! n2
-- WORKING ON n4
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
  FR.6 ? f0; f1; f2  -- f6 FROM P6
  V.COS:=T.COS[THETA04]
  V.SIN:=T.SIN[THETA04]
  -- CALCULATION OF RP(I+1,I) X RF(I+1)-( R(I+1,I)RP(I)) X RF(I+1)
  B0:= V.COS * L
  B1:= V.SIN * L
  C0:= n0 - (B1*f2)
  C1:= n1 - (B0*f2)
  C2:= n2 + ((B0*f1)+(B1*f0))
  -- CALCULATION OF R(I, I+1)*( R(I+1, I)n(I+1) + RP(I+1,I) X RF(I+1)
  B0:= (V.COS * C0) - (V.SIN * C1)
  B1:= (V.SIN * C0) + (V.COS * C1)
  B2:= C2
  FR.8 ? nI0; nI1; nI2
  n0:= B0 + nI0
  n1:= B1 + nI1
  n2:= B2 + nI2
  TO.1 ! n2
-- WORKING ON n3
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
```

```
SEQ
  FR.6 ? f0; f1; f2  -- f6 FROM P6
  V.COS:=T.COS[THETA03]
  V.SIN:=T.SIN[THETA03]
  -- CALCULATION OF RP(I+1,I) X RF(I+1)-( R(I+1,I)RP(I)) X RF(I+1)
  B0:= V.COS * L
  B1:= V.SIN * L
  C0:= n0 - (B1*f2)
  C1:= n1 - (B0*f2)
  C2:= (B0*f1)+(B1*f0)
  C2:= C2 + n2
  -- CALCULATION OF R(I, I+1)*( R(I+1, I)n(I+1) + RP(I+1,I) X RF(I+1)
  B0:= (V.COS * C0) - (V.SIN * C1)
  B1:= (V.SIN * C0) + (V.COS * C1)
  B2:=  C2
  FR.8 ? nI0; nI1; nI2
  n0:= B0 + nI0
  n1:= B1 + nI1
  n2:= B2 + nI2
  TO.1 ! n2
-- WORKING ON n2
VAL RF02 IS Z:
VAL RF12 IS Z:
VAL RF20 IS Z:
VAL RF21 IS Z:
VAL RF22 IS ONE:
SEQ
  FR.6 ? f0; f1; f2  -- f6 FROM P6
  V.COS:=T.COS[THETA02]
  V.SIN:=T.SIN[THETA02]
  -- CALCULATION OF RP(I+1,I) X RF(I+1)-( R(I+1,I)RP(I)) X RF(I+1)
  B0:= V.COS * L
  B1:= V.SIN * L
  C0:= n0 - (B1*f2)
  C1:= n1 - (B0*f2)
  C2:= (B0*f1)+(B1*f0)
  C2:= C2 + n2
  -- CALCULATION OF R(I, I+1)*( R(I+1, I)n(I+1) + RP(I+1,I) X RF(I+1)
  B0:= (V.COS * C0) - (V.SIN * C1)
```

```
    B1:- (V.SIN * C0) + (V.COS * C1)
    B2:- C2
    FR.8 ? nI0; nI1; nI2
    n0:- B0 + nI0
    n1:- B1 + nI1
    n2:- B2 + nI2
    TO.1 ! n2
  -- WORKING ON n1
  VAL RF02 IS Z:
  VAL RF12 IS Z:
  VAL RF20 IS Z:
  VAL RF21 IS Z:
  VAL RF22 IS ONE:
  SEQ
    FR.6 ? f0; f1; f2  -- f6 FROM P6
    V.COS:-T.COS[THETA01]
    V.SIN:-T.SIN[THETA01]
    -- CALCULATION OF RP(I+1,I) X RF(I+1)-( R(I+1,I)RP(I)) X RF(I+1)
    B0:- V.COS * L
    B1:- V.SIN * L
    C0:- n0 - (B1*f2)
    C1:- n1 - (B0*f2)
    C2:- (B0*f1)+(B1*f0)
    C2:- C2 + n2
    -- CALCULATION OF R(I, I+1)*( R(I+1, I)n(I+1) + RP(I+1,I) X RF(I+1)
    B0:- (V.COS * C0) - (V.SIN * C1)
    B1:- (V.SIN * C0) + (V.COS * C1)
    B2:- C2
    FR.8 ? nI0; nI1; nI2
    n0:- B0 + nI0
    n1:- B1 + nI1
    n2:- B2 + nI2
    TO.1 ! n2
:
#USE LIBTH4
PROC EIGHT(CHAN OF ME3R FR.0, CHAN OF ME3R FR.6,
       CHAN OF ME3R FR.7, CHAN OF ME3R TO.7)
 INT I, J, K:
 [6]INT THETA0:
```

```
 REAL32 W0, W1, W2, W.0,  W.1,  W.2, N0, N1, N2, N0.I, N1.I, N2.I, B0, B1,
B2:
 REAL32 C0, C1, C2, F0, F1, F2, n0, n1, n2:
 REAL32 N50, N51, N52, N40, N41, N42, N30, N31, N32, N20, N21, N22,
    N10, N11, N12, N00, N01, N02:
 SEQ
  VAL RIR0 IS Z:
  VAL RIR1 IS Z:
  VAL RIR2 IS Z:
  VAL RIR3 IS Z:
  VAL RIR4 IS INER1:
  VAL RIR5 IS Z:
  VAL RIR6 IS Z:
  VAL RIR7 IS Z:
  VAL RIR8 IS INER1:
  SEQ
   FR.0 ? W0; W1; W2
   -- CALCULATION OF RN.I
   -- RW X [(RIR)(RW)]
   C0:= (RIR0 * W0)+ (RIR1 * W1)
   C1:= (RIR3 * W0)+ (RIR4 * W1)
   C2:= (RIR6 * W0)+ (RIR7 * W1)
   B0:= C0+(RIR2 * W2)
   B1:= C1+(RIR5 * W2)
   B2:= C2+(RIR8 * W2)
   N00:= (W1 * B2)- (W2 * B1)
   N01:= (W2 * B0)- (W0 * B2)
   N02:= (W0 * B1)- (W1 * B0)
   -- RECEIVE PARTIAL RN FROM P7
   FR.7 ? N0; N1; N2
   -- ADD TWO PARTIAL RN
   N00:= N00 + N0
   N01:= N01 + N1
   N02:= N02 + N2
  VAL RIR0 IS Z:
  VAL RIR1 IS Z:
  VAL RIR2 IS Z:
  VAL RIR3 IS Z:
  VAL RIR4 IS INER:
```

```
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER:
SEQ
  FR.0 ? W0; W1; W2
  -- CALCULATION OF RN.I
  -- RW X [(RIR)(RW)]
  C0:= (RIR0 * W0)+ (RIR1 * W1)
  C1:= (RIR3 * W0)+ (RIR4 * W1)
  C2:= (RIR6 * W0)+ (RIR7 * W1)
  B0:= C0+(RIR2 * W2)
  B1:= C1+(RIR5 * W2)
  B2:= C2+(RIR8 * W2)
  N10:= (W1 * B2)- (W2 * B1)
  N11:= (W2 * B0)- (W0 * B2)
  N12:= (W0 * B1)- (W1 * B0)
  -- RECEIVE PARTIAL RN FROM P7
  FR.7 ? N0; N1; N2
  -- ADD TWO PARTIAL RN
  N10:= N10 + N0
  N11:= N11 + N1
  N12:= N12 + N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER1:
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER1:
SEQ
  FR.0 ? W0; W1; W2
  -- CALCULATION OF RN.I
  -- RW X [(RIR)(RW)]
  C0:= (RIR0 * W0)+ (RIR1 * W1)
  C1:= (RIR3 * W0)+ (RIR4 * W1)
  C2:= (RIR6 * W0)+ (RIR7 * W1)
```

```
 B0:- C0+(RIR2 * W2)
 B1:- C1+(RIR5 * W2)
 B2:- C2+(RIR8 * W2)
 N20:- (W1 * B2)- (W2 * B1)
 N21:- (W2 * B0)- (W0 * B2)
 N22:- (W0 * B1)- (W1 * B0)
 -- RECEIVE PARTIAL RN FROM P7
 FR.7 ? N0; N1; N2
 -- ADD TWO PARTIAL RN
 N20:- N20 + N0
 N21:- N21 + N1
 N22:- N22 + N2
VAL RIR0 IS Z:
VAL RIR1 IS Z:
VAL RIR2 IS Z:
VAL RIR3 IS Z:
VAL RIR4 IS INER1:
VAL RIR5 IS Z:
VAL RIR6 IS Z:
VAL RIR7 IS Z:
VAL RIR8 IS INER1:
SEQ
  FR.0 ? W0; W1; W2
  -- CALCULATION OF RN.I
  -- RW X [(RIR)(RW)]
  C0:- (RIR0 * W0)+ (RIR1 * W1)
  C1:- (RIR3 * W0)+ (RIR4 * W1)
  C2:- (RIR6 * W0)+ (RIR7 * W1)
  B0:- C0+(RIR2 * W2)
  B1:- C1+(RIR5 * W2)
  B2:- C2+(RIR8 * W2)
  N30:- (W1 * B2)- (W2 * B1)
  N31:- (W2 * B0)- (W0 * B2)
  N32:- (W0 * B1)- (W1 * B0)
  -- RECEIVE PARTIAL RN FROM P7
  FR.7 ? N0; N1; N2
  -- ADD TWO PARTIAL RN
  N30:- N30 + N0
  N31:- N31 + N1
```

```
  N32:= N32 + N2
 VAL RIR0 IS Z:
 VAL RIR1 IS Z:
 VAL RIR2 IS Z:
 VAL RIR3 IS Z:
 VAL RIR4 IS INER1:
 VAL RIR5 IS Z:
 VAL RIR6 IS Z:
 VAL RIR7 IS Z:
 VAL RIR8 IS INER1:
 SEQ
  FR.0 ? W0; W1; W2
  -- CALCULATION OF RN.I
  -- RW X [(RIR)(RW)]
  C0:= (RIR0 * W0)+ (RIR1 * W1)
  C1:= (RIR3 * W0)+ (RIR4 * W1)
  C2:= (RIR6 * W0)+ (RIR7 * W1)
  B0:= C0+(RIR2 * W2)
  B1:= C1+(RIR5 * W2)
  B2:= C2+(RIR8 * W2)
  N40:= (W1 * B2)- (W2 * B1)
  N41:= (W2 * B0)- (W0 * B2)
  N42:= (W0 * B1)- (W1 * B0)
  -- RECEIVE PARTIAL RN FROM P7
  FR.7 ? N0; N1; N2
  -- ADD TWO PARTIAL RN
  N40:= N40 + N0
  N41:= N41 + N1
  N42:= N42 + N2
 VAL RIR0 IS Z:
 VAL RIR1 IS Z:
 VAL RIR2 IS Z:
 VAL RIR3 IS Z:
 VAL RIR4 IS INER1:
 VAL RIR5 IS Z:
 VAL RIR6 IS Z:
 VAL RIR7 IS Z:
 VAL RIR8 IS INER1:
 SEQ
```

```
 FR.0 ? W0; W1; W2
 -- CALCULATION OF RN.I
 -- RW X [(RIR)(RW)]
 C0:= (RIR0 * W0)+ (RIR1 * W1)
 C1:= (RIR3 * W0)+ (RIR4 * W1)
 C2:= (RIR6 * W0)+ (RIR7 * W1)
 B0:= C0+(RIR2 * W2)
 B1:= C1+(RIR5 * W2)
 B2:= C2+(RIR8 * W2)
 N50:= (W1 * B2)- (W2 * B1)
 N51:= (W2 * B0)- (W0 * B2)
 N52:= (W0 * B1)- (W1 * B0)
 -- RECEIVE PARTIAL RN FROM P7
 FR.7 ? N0; N1; N2
 -- ADD TWO PARTIAL RN
 N50:= N50 + N0
 N51:= N51 + N1
 N52:= N52 + N2
-- WORKING ON n6.I
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
 FR.6 ? F0; F1; F2
 -- CALCULATION OF Rn.I=(RP + RS) X (RF) + RN
 C0:= RP0 + RS0
 C1:= RP1 + RS1
 C2:= RP2 + RS2
 B0:= (C1 * F2) - (C2 * F1)
 B1:= (C2 * F0) - (C0 * F2)
 B2:= (C0 * F1) - (C1 * F0)
 n0:= B0 + N50
 n1:= B1 + N51
 n2:= B2 + N52
 TO.7 ! n0; n1; n2
-- WORKING ON n5.I
```

```
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
 FR.6 ? F0; F1; F2
 -- CALCULATION OF Rn.I-(RP + RS) X (RF) + RN
 C0:= RP0 + RS0
 C1:= RP1 + RS1
 C2:= RP2 + RS2
 B0:= (C1 * F2) - (C2 * F1)
 B1:= (C2 * F0) - (C0 * F2)
 B2:= (C0 * F1) - (C1 * F0)
 n0:= B0 + N40
 n1:= B1 + N41
 n2:= B2 + N42
 TO.7 ! n0; n1; n2
-- WORKING ON n4.I
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
 FR.6 ? F0; F1; F2
 -- CALCULATION OF Rn.I-(RP + RS) X (RF) + RN
 C0:= RP0 + RS0
 C1:= RP1 + RS1
 C2:= RP2 + RS2
 B0:= (C1 * F2) - (C2 * F1)
 B1:= (C2 * F0) - (C0 * F2)
 B2:= (C0 * F1) - (C1 * F0)
 n0:= B0 + N30
 n1:= B1 + N31
 n2:= B2 + N32
 TO.7 ! n0; n1; n2
```

```
-- WORKING ON n3.I
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
  FR.6 ? F0; F1; F2
  -- CALCULATION OF Rn.I-(RP + RS) X (RF) + RN
  C0:= RP0 + RS0
  C1:= RP1 + RS1
  C2:= RP2 + RS2
  B0:= (C1 * F2) - (C2 * F1)
  B1:= (C2 * F0) - (C0 * F2)
  B2:= (C0 * F1) - (C1 * F0)
  n0:= B0 + N20
  n1:= B1 + N21
  n2:= B2 + N22
  TO.7 ! n0; n1; n2
-- WORKING ON n2.I
VAL RP0 IS L:
VAL RP1 IS Z:
VAL RP2 IS Z:
VAL RS0 IS S:
VAL RS1 IS Z:
VAL RS2 IS Z:
SEQ
  FR.6 ? F0; F1; F2
  -- CALCULATION OF Rn.I-(RP + RS) X (RF) + RN
  C0:= RP0 + RS0
  C1:= RP1 + RS1
  C2:= RP2 + RS2
  B0:= (C1 * F2) - (C2 * F1)
  B1:= (C2 * F0) - (C0 * F2)
  B2:= (C0 * F1) - (C1 * F0)
  n0:= B0 + N10
  n1:= B1 + N11
  n2:= B2 + N12
```

```
    TO.7 ! n0; n1; n2
   -- WORKING ON n1.I
   VAL RP0 IS L:
   VAL RP1 IS Z:
   VAL RP2 IS Z:
   VAL RS0 IS S:
   VAL RS1 IS Z:
   VAL RS2 IS Z:
   SEQ
    FR.6 ? F0; F1; F2
    -- CALCULATION OF Rn.I-(RP + RS) X (RF) + RN
    C0:= RP0 + RS0
    C1:= RP1 + RS1
    C2:= RP2 + RS2
    B0:= (C1 * F2) - (C2 * F1)
    B1:= (C2 * F0) - (C0 * F2)
    B2:= (C0 * F1) - (C1 * F0)
    n0:= B0 + N00
    n1:= B1 + N01
    n2:= B2 + N02
    TO.7 ! n0; n1; n2
:
CHAN OF MECC FH.0:
CHAN OF INT F1.6:
CHAN OF REAL32 F1.0, F7.1:
CHAN OF METT F0.H:
CHAN OF ME3RTHE F0.1, F0.2:
CHAN OF ME3R F0.8, F2.1, F3.4, F3.5, F4.3, F4.5, F5.6, F6.7, F6.8, F7.8, F8.7:
CHAN OF ME3RTHE0 F1.7, F2.3, F2.4:
PLACED PAR
 PROCESSOR 0 T4
  PLACE FH.0 AT L1I:
  PLACE F1.0 AT L2I:
  PLACE F0.H AT L1O:
  PLACE F0.1 AT L2O:
  PLACE F0.2 AT L0O:
  PLACE F0.8 AT L3O:
  ZEROO(FH.0, F1.0, F0.H, F0.1, F0.2, F0.8)
 PROCESSOR 1 T8
```

```
  PLACE F0.1 AT L1I:
  PLACE F2.1 AT L2I:
  PLACE F7.1 AT L3I:
  PLACE F1.0 AT L1O:
  PLACE F1.6 AT L0O:
  PLACE F1.7 AT L3O:
  FIRST(F0.1, F2.1, F7.1, F1.0, F1.6, F1.7)
PROCESSOR 2 T8
  PLACE F0.2 AT L0I:
  PLACE F2.1 AT L1O:
  PLACE F2.3 AT L2O:
  PLACE F2.4 AT L3O:
  SECON(F0.2, F2.1, F2.3, F2.4)
PROCESSOR 3 T8
  PLACE F2.3 AT L1I:
  PLACE F4.3 AT L2I:
  PLACE F3.4 AT L2O:
  PLACE F3.5 AT L0O:
  THIRD(F2.3, F4.3, F3.4, F3.5)
PROCESSOR 4 T8
  PLACE F2.4 AT L3I:
  PLACE F3.4 AT L1I:
  PLACE F4.3 AT L1O:
  PLACE F4.5 AT L2O:
  FORTH(F2.4, F3.4, F4.3, F4.5)
PROCESSOR 5 T8
  PLACE F3.5 AT L0I:
  PLACE F4.5 AT L1I:
  PLACE F5.6 AT L2O:
  FIFTH(F3.5, F4.5, F5.6)
PROCESSOR 6 T8
  PLACE F1.6 AT L0I:
  PLACE F5.6 AT L1I:
  PLACE F6.7 AT L2O:
  PLACE F6.8 AT L3O:
  SIXTH(F1.6, F5.6, F6.7, F6.8)
PROCESSOR 7 T8
  PLACE F1.7 AT L3I:
  PLACE F6.7 AT L1I:
```

```
  PLACE F8.7 AT L2I:
  PLACE F7.1 AT L3O:
  PLACE F7.8 AT L2O:
  SEVEN(F1.7, F6.7, F8.7, F7.1, F7.8)
PROCESSOR 8 T8
  PLACE F0.8 AT L0I:
  PLACE F6.8 AT L3I:
  PLACE F7.8 AT L1I:
  PLACE F8.7 AT L1O:
  EIGHT(F0.8, F6.8, F7.8, F8.7)
```